scilpy Documentation

SCIL

Apr 13, 2023

Scilpy

1 Modules 1

2 Scripts 51

3 Instructions for streamlines registration/transformation 199

Bibliography 201

Python Module Index 203

Index 205

CHAPTER 1

Modules

1.1 scilpy.gradientsampling package

1.1.1 scilpy.gradientsampling.gen_gradient_sampling module

scilpy.gradientsampling.gen_gradient_sampling.**generate_gradient_sampling**(*nb_samples*, *verbose=1*)

Wrapper code to generate gradient sampling from Caruyer's multiple_shell_energy.py

Generate the bvecs of a multiple shell gradient sampling using generalized Jones electrostatic repulsion.

Parameters

- **nb_samples** (*list*) – number of samples for each shell, starting from lowest.
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations*) –

Returns

- **points** (*numpy.array*) – bvecs normalized to 1.
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.

1.1.2 scilpy.gradientsampling.multiple_shell_energy module

scilpy.gradientsampling.multiple_shell_energy.**compute_weights**(*nb_shells*, *nb_points_per_shell*, *shell_groups*, *alphas*)

Computes the weights array from a set of shell groups to couple, and coupling weights.

Parameters

- **nb_shells** (*int*) – Number of shells

- **nb_points_per_shell** (*list of ints*) – Number of points per shell
- **shell_groups** (*tuple*) – tuple listing the groups of shells as lists of indices
- **alphas** (*list*) – list of weights per group of shells

Returns **weights** – weigths for each group of shells

Return type nd.ndarray

scilpy.gradientsampling.multiple_shell_energy.**cost**(*bvecs*, *S*, *Ks*, *weights*)

Objective function for multiple-shell energy.

Parameters

- **bvecs** (*array-like shape (N * 3,)*) –
- **S** (*int*) – Number of shells.
- **Ks** (*list of ints, len(Ks) = S. Number of points per shell.*) –
- **weights** (*array-like, shape (S, S)*) – Weighting parameter, control coupling between shells and how this balances.

Returns **electrostatic_repulsion** – sum of all interactions between any two vectors.

Return type float

scilpy.gradientsampling.multiple_shell_energy.**electrostatic_repulsion**(*bvecs*, *weight_matrix*, *alpha=1.0*)

Electrostatic-repulsion objective function. The alpha parameter controls the power repulsion (energy varies as $1 / ralpha$).

Parameters

- **bvecs** (*array-like shape (N * 3,)*) – Vectors.
- **weight_matrix** (*array-like, shape (N, N)*) – The contribution weight of each pair of points.
- **alpha** (*float*) – Controls the power of the repulsion. Default is 1.0

Returns **energy** – sum of all interactions between any two vectors.

Return type float

scilpy.gradientsampling.multiple_shell_energy.**equality_constraints**(*bvecs*, **args*)

Spherical equality constraint. Returns 0 if bvecs lies on the unit sphere.

Parameters **bvecs** (*array-like shape (N * 3)*) –

Returns **array shape (N,)**

Return type Difference between squared vector norms and 1.

scilpy.gradientsampling.multiple_shell_energy.**grad_cost**(*bvecs*, *S*, *Ks*, *weights*)

gradient of the objective function for multiple shells sampling.

Parameters

- **bvecs** (*array-like shape (N * 3,)*) –
- **S** (*int*) – number of shells
- **Ks** (*list of ints*) – len(Ks) = S. Number of points per shell.

- **weights** (*array-like, shape (S, S)*) – weighting parameter, control coupling between shells and how this balances.

Returns **grad_electrostatic_repulsion** – gradient of the objective function

Return type float

scilpy.gradientsampling.multiple_shell_energy.**grad_electrostatic_repulsion**(*bvecs*, *weight_matrix*, *alpha=1.0*)

1st-order derivative of electrostatic-like repulsion energy.

Parameters

- **bvecs** (*array-like shape (N * 3,)*) – Vectors.
- **weight_matrix** (*array-like, shape (N, N)*) – The contribution weight of each pair of points.
- **alpha** (*floating-point. controls the power of the repulsion. Default is 1.0*) –

Returns **grad** – gradient of the objective function

Return type numpy.ndarray

scilpy.gradientsampling.multiple_shell_energy.**grad_equality_constraints**(*bvecs*, **args*)

Return normals to the surface constraint (wich corresponds to the gradient of the implicit function).

Parameters **bvecs** (*array-like shape (N * 3)*) –

Returns

- *array shape (N, N * 3). grad[i, j] contains*
- *$partial f_i / partial x_j$*

scilpy.gradientsampling.multiple_shell_energy.**multiple_shell**(*nb_shells*, *nb_points_per_shell*, *weights*, *max_iter=100*, *verbose=2*)

Creates a set of sampling directions on the desired number of shells.

Parameters

- **nb_shells** (*the number of shells*) –
- **nb_points_per_shell** (*list, shape (nb_shells,)*) – A list of integers containing the number of points on each shell.
- **weights** (*array-like, shape (S, S)*) – weighting parameter, control coupling between shells and how this balances.
- **max_iter** (*int*) – Maximum number of interations

Returns **bvecs** – The points are stored by shell.

Return type array shape (K, 3) where K is the total number of points

scilpy.gradientsampling.multiple_shell_energy.**random_uniform_on_sphere**(*K*)

Creates a set of K pseudo-random unit vectors, following a uniform distribution on the sphere.

Parameters **K** (*int*) – Number of vectors

Returns **bvecs** – pseudo-random unit vector

Return type nd.array

scilpy.gradientsampling.multiple_shell_energy.**write_multiple_shells**(*bvecs*, *nb_shells*, *nb_points_per_shell*, *filename*)

Export multiple shells to text file.

Parameters

- **bvecs** (*array-like shape (K, 3)*) – vectors
- **nb_shells** (*int*) – Number of shells
- **nb_points_per_shell** (*array-like shape (nb_shells,)*) – A list of integers containing the number of points on each shell.
- **filename** (*str*) – output filename

1.1.3 scilpy.gradientsampling.optimize_gradient_sampling module

scilpy.gradientsampling.optimize_gradient_sampling.**add_b0s**(*points*, *shell_idx*, *b0_every=10*, *finish_b0=False*, *verbose=1*)

Add interleaved b0s to gradient sampling.

Parameters

- **points** (*numpy.array, bvecs normalized to 1.*) –
- **shell_idx** (*numpy.array, Shell index for bvecs in points.*) –
- **b0_every** (*integer, final gradient sampling will have a b0 every b0_every*) – samples
- **finish_b0** (*boolean, Option to add a b0 as last sample.*) –
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns

- **points** (*numpy.array*) – bvecs normalized to 1.
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.

scilpy.gradientsampling.optimize_gradient_sampling.**add_bvalue_b0**(*bvals*, *b0_value=0.0*)

Add the b0 value to the bvals list.

Parameters

- **bvals** (*list*) – bvals of the non-b0 shells.
- **b0_value** (*float*) – bvals of the b0s
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns **bvals** – bvals of the shells and b0s.

Return type list

scilpy.gradientsampling.optimize_gradient_sampling.**compute_bvalue_lin_b**(*bmin=0.0*, *bmax=3000.0*, *nb_of_b_inside=2*, *exclude_bmin=True*, *verbose=1*)

Compute bvals linearly distributed in b-value in the interval [bmin, bmax].

Parameters

- **bmin** (*float*) – Minimum b-value, lower b-value bounds.
- **bmax** (*float*) – Maximum b-value, upper b-value bounds.
- **nb_of_b_inside** (*int*) – number of b-value excluding bmin and bmax.
- **exclude_bmin** (*boolean*) – exclude bmin from the interval, useful if bmin = 0.0.
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns **bvals** – increasing bvals.

Return type list

scilpy.gradientsampling.optimize_gradient_sampling.**compute_bvalue_lin_q**(*bmin=0.0*, *bmax=3000.0*, *nb_of_b_inside=2*, *exclude_bmin=True*, *verbose=1*)

Compute bvals linearly distributed in q-value in the interval [bmin, bmax].

Parameters

- **bmin** (*float*) – Minimum b-value, lower b-value bounds.
- **bmax** (*float*) – Maximum b-value, upper b-value bounds.
- **nb_of_b_inside** (*int*) – number of b-value excluding bmin and bmax.
- **exclude_bmin** (*bool*) – exclude bmin from the interval, useful if bmin = 0.0.
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns **bvals** – increasing bvals.

Return type list

scilpy.gradientsampling.optimize_gradient_sampling.**compute_ks_from_shell_idx**(*shell_idx*)

Recover number of points per shell from point-wise shell index.

Parameters **shell_idx** (*numpy.array*) – Shell index of gradient sampling.

Returns **Ks** – number of samples for each shell, starting from lowest.

Return type list

scilpy.gradientsampling.optimize_gradient_sampling.**compute_min_duty_cycle_bruteforce**(*points*, *shell_id bvals*, *ker_size Niter=1 verbose=1*, *plotting=Fa rand_se*

Optimize the ordering of non-b0s sample to optimize gradient duty-cycle.

Philips scanner (and other) will find the peak power requirements with its duty cycle model (this is an approximation) and increase the TR accordingly to the hardware needs. This minimize this effects by:

1) Randomly permuting the non-b0s samples
2) Finding the peak X, Y, and Z amplitude with a sliding-window
3) Compute peak power needed as max(peak_x, peak_y, peak_z)
4) Keeps the permutation yielding the lowest peak power

Parameters

- **points** (*numpy.array*) – bvecs normalized to 1
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.
- **bvals** (*list*) – increasing bvals, b0 last.
- **ker_size** (*int*) – kernel size for the sliding window.
- **Niter** (*int*) – number of bruteforce iterations.
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –
- **plotting** (*bool*) – plot the energy at each iteration.
- **rand_seed** (*int*) – seed for the random permutations.

Returns

- **points** (*numpy.array*) – bvecs normalized to 1.
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.

scilpy.gradientsampling.optimize_gradient_sampling.**compute_peak_power**(*q_scheme*, *ker_size=10*)

Parameters

- **q_scheme** (*nd.array*) – Scheme of acquisition.
- **ker_size** (*int*) – Kernel size (default=10).

Returns

Return type Max peak power from q_scheme.

scilpy.gradientsampling.optimize_gradient_sampling.**correct_b0s_philips**(*points*, *shell_idx*, *verbose=1*)

Replace the [0.0, 0.0, 0.0] value of b0s bvecs by existing bvecs in the gradient sampling.

This is useful because Recon 1.0 of Philips allocates memory proportional to (total nb. of diff. bvals) x (total nb. diff. bvecs) and we can't leave multiple b0s with b-vector [0.0, 0.0, 0.0] and b-value 0 because (b-vector, b-value) pairs have to be unique.

Parameters

- **points** (*numpy.array*) – bvecs normalized to 1
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns

- **points** (*numpy.array*) – bvecs normalized to 1
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points

scilpy.gradientsampling.optimize_gradient_sampling.**swap_sampling_eddy**(*points*, *shell_idx*, *verbose=1*)

Optimize the bvecs of fixed multi-shell gradient sampling for eddy currents correction (fsl EDDY).

Bruteforce approach to maximally spread the bvec, shell per shell.

For each shell:

For each vector:

1) find the closest neighbor,
2) flips it,
3) if global system energy is better, keep it flipped

repeat until convergence.

Parameters

- **points** (*numpy.array, bvecs normalized to 1.*) –
- **shell_idx** (*numpy.array, Shell index for bvecs in points.*) –
- **verbose** (*0 = silent, 1 = summary upon completion, 2 = print iterations.*) –

Returns

- **points** (*numpy.array, bvecs normalized to 1.*)
- **shell_idx** (*numpy.array, Shell index for bvecs in points.*)

1.1.4 scilpy.gradientsampling.save_gradient_sampling module

scilpy.gradientsampling.save_gradient_sampling.**save_gradient_sampling_fsl**(*points*, *shell_idx*, *bvals*, *filename_bval*, *filename_bvec*)

Save table gradient (FSL format)

Parameters

- **points** (*numpy.array*) – bvecs normalized to 1.
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.
- **bvals** (*numpy.array*) –
- **filename_bval** (*str*) – output bval filename.
- **filename_bvec** (*str*) – output bvec filename.
- **------** –

scilpy.gradientsampling.save_gradient_sampling.**save_gradient_sampling_mrtrix**(*points*, *shell_idx*, *bvals*, *filename*)

Save table gradient (MRtrix format)

Parameters

- **points** (*numpy.array*) – bvecs normalized to 1.
- **shell_idx** (*numpy.array*) – Shell index for bvecs in points.
- **bvals** (*numpy.array*) –
- **filename** (*str*) – output file name
- **------** –

1.2 scilpy.image package

1.2.1 scilpy.image.operations module

Utility operations provided for scil_image_math.py and scil_connectivity_math.py They basically act as wrappers around numpy to avoid installing MRtrix/FSL to apply simple operations on nibabel images or numpy arrays.

scilpy.image.operations.**absolute_value**(*input_list*, *ref_img*)

absolute_value: IMG All negative values will become positive.

scilpy.image.operations.**addition**(*input_list*, *ref_img*)

addition: IMGs Add multiple images together.

scilpy.image.operations.**around**(*input_list*, *ref_img*)

round: IMG Round all decimal values to the closest integer.

scilpy.image.operations.**base_10_log**(*input_list*, *ref_img*)

log_10: IMG Apply a log (base 10) to all non zeros values of an image.

scilpy.image.operations.**ceil**(*input_list*, *ref_img*)

ceil: IMG Ceil all decimal values to the next integer.

scilpy.image.operations.**closing**(*input_list*, *ref_img*)

closing: IMG, VALUE Binary morphological operation, dilation followed by an erosion.

scilpy.image.operations.**concatenate**(*input_list*, *ref_img*)

concatenate: IMGs Concatenate a list of 3D and 4D images into a single 4D image.

scilpy.image.operations.**convert**(*input_list*, *ref_img*)

convert: IMG Perform no operation, but simply change the data type.

scilpy.image.operations.**difference**(*input_list*, *ref_img*)

difference: IMG_1 IMG_2 Operation on binary image to keep voxels from the first file that are not in the second file (non-zeros).

scilpy.image.operations.**dilation**(*input_list*, *ref_img*)

dilation: IMG, VALUE Binary morphological operation to spatially extend the values of an image to their neighbors. VALUE is in voxels.

scilpy.image.operations.**division**(*input_list*, *ref_img*)

division: IMG_1 IMG_2 Divide first image by the second (danger of underflow and overflow) Ignore zeros values, excluded from the operation.

scilpy.image.operations.**erosion**(*input_list*, *ref_img*)

erosion: IMG, VALUE Binary morphological operation to spatially shrink the volume contained in a binary image. VALUE is in voxels.

scilpy.image.operations.**floor**(*input_list*, *ref_img*)

floor: IMG Floor all decimal values to the previous integer.

scilpy.image.operations.**gaussian_blur**(*input_list*, *ref_img*)

blur: IMG, VALUE Apply a gaussian blur to a single image.

scilpy.image.operations.**get_array_ops**()
Get a dictionary of all functions relating to array operations

scilpy.image.operations.**get_image_ops**()
Get a dictionary of all functions relating to image operations

scilpy.image.operations.**get_operations_doc**(*ops: dict*)
From a dictionary mapping operation names to functions, fetch and join all documentations, using the provided names.

scilpy.image.operations.**intersection**(*input_list*, *ref_img*)

intersection: IMGs Operation on binary image to keep the voxels, that are non-zero, are present in all files.

scilpy.image.operations.**invert**(*input_list*, *ref_img*)

invert: IMG Operation on binary image to interchange 0s and 1s in a binary mask.

scilpy.image.operations.**lower_clip**(*input_list*, *ref_img*)

lower_clip: IMG THRESHOLD All values below the threshold will be set to threshold.

scilpy.image.operations.**lower_threshold**(*input_list*, *ref_img*)

lower_threshold: IMG THRESHOLD All values below the threshold will be set to zero. All values above the threshold will be set to one.

scilpy.image.operations.**lower_threshold_eq**(*input_list*, *ref_img*)

lower_threshold_eq: IMG THRESHOLD All values below the threshold will be set to zero. All values above or equal the threshold will be set to one.

scilpy.image.operations.**mean**(*input_list*, *ref_img*)

mean: IMGs Compute the mean of images. If a single 4D image is provided, average along the last dimension.

scilpy.image.operations.**multiplication**(*input_list*, *ref_img*)

multiplication: IMGs Multiply multiple images together (danger of underflow and overflow)

scilpy.image.operations.**natural_log**(*input_list*, *ref_img*)

log_e: IMG Apply a natural log to all non zeros values of an image.

scilpy.image.operations.**normalize_max**(*input_list*, *ref_img*)

normalize_max: IMG Normalize the image so the maximum value is one.

scilpy.image.operations.**normalize_sum**(*input_list*, *ref_img*)

normalize_sum: IMG Normalize the image so the sum of all values is one.

scilpy.image.operations.**opening**(*input_list*, *ref_img*)

opening: IMG, VALUE Binary morphological operation, erosion followed by a dilation.

scilpy.image.operations.**std**(*input_list*, *ref_img*)

std: IMGs Compute the standard deviation average of multiple images. If a single 4D image is provided, compute the STD along the last dimension.

scilpy.image.operations.**subtraction**(*input_list*, *ref_img*)

subtraction: IMG_1 IMG_2 Subtract first image by the second (IMG_1 - IMG_2).

scilpy.image.operations.**union**(*input_list*, *ref_img*)

union: IMGs Operation on binary image to keep voxels, that are non-zero, in at least one file.

scilpy.image.operations.**upper_clip**(*input_list*, *ref_img*)

upper_clip: IMG THRESHOLD All values above the threshold will be set to threshold.

scilpy.image.operations.**upper_threshold**(*input_list*, *ref_img*)

upper_threshold: IMG THRESHOLD All values below the threshold will be set to one. All values above the threshold will be set to zero. Equivalent to lower_threshold followed by an inversion.

scilpy.image.operations.**upper_threshold_eq**(*input_list*, *ref_img*)

upper_threshold_eq: IMG THRESHOLD All values below or equal the threshold will be set to one. All values above the threshold will be set to zero. Equivalent to lower_threshold followed by an inversion.

1.2.2 scilpy.image.resample_volume module

scilpy.image.resample_volume.**resample_volume**(*img*, *ref=None*, *res=None*, *iso_min=False*, *zoom=None*, *interp='lin'*, *enforce_dimensions=False*)

Function to resample a dataset to match the resolution of another reference dataset or to the resolution specified as in argument. One of the following options must be chosen: ref, res or iso_min.

Parameters

- **img** (*nib.Nifti1Image*) – Image to resample.
- **ref** (*nib.Nifti1Image*) – Reference volume to resample to. This method is used only if ref is not None. (default: None)

- **res** (*tuple, shape (3,) or int, optional*) – Resolution to resample to. If the value it is set to is Y, it will resample to an isotropic resolution of Y x Y x Y. This method is used only if res is not None. (default: None)
- **iso_min** (*bool, optional*) – If true, resample the volume to R x R x R with R being the smallest current voxel dimension. If false, this method is not used.
- **zoom** (*tuple, shape (3,) or float, optional*) – Set the zoom property of the image at the value specified.
- **interp** (*str, optional*) – Interpolation mode. 'nn' = nearest neighbour, 'lin' = linear, 'quad' = quadratic, 'cubic' = cubic. (Default: linear)
- **enforce_dimensions** (*bool, optional*) – If True, enforce the reference volume dimension (only if res is not None). (Default = False)

Returns **resampled_image** – Resampled image.

Return type nib.Nifti1Image

1.2.3 scilpy.image.reslice module

scilpy.image.reslice.**reslice**(*data*, *affine*, *zooms*, *new_zooms*, *order=1*, *mode='constant'*, *cval=0*, *num_processes=1*)

Reslice data with new voxel resolution defined by new_zooms

Parameters

- **data** (*array, shape (I,J,K) or (I,J,K,N)*) – 3d volume or 4d volume with datasets
- **affine** (*array, shape (4,4)*) – mapping from voxel coordinates to world coordinates
- **zooms** (*tuple, shape (3,)*) – voxel size for (i,j,k) dimensions
- **new_zooms** (*tuple, shape (3,)*) – new voxel size for (i,j,k) after resampling
- **order** (*int, from 0 to 5*) – order of interpolation for resampling/reslicing, 0 nearest interpolation, 1 trilinear etc.. if you don't want any smoothing 0 is the option you need.
- **mode** (*string ('constant', 'nearest', 'reflect' or 'wrap')*) – Points outside the boundaries of the input are filled according to the given mode.
- **cval** (*float*) – Value used for points outside the boundaries of the input if mode='constant'.
- **num_processes** (*int*) – Split the calculation to a pool of children processes. This only applies to 4D *data* arrays. If a positive integer then it defines the size of the multiprocessing pool that will be used. If 0, then the size of the pool will equal the number of cores available.

Returns

- **data2** (*array, shape (I,J,K) or (I,J,K,N)*) – datasets resampled into isotropic voxel size
- **affine2** (*array, shape (4,4)*) – new affine for the resampled image

Examples

```
>>> from dipy.io.image import load_nifti
>>> from dipy.align.reslice import reslice
>>> from dipy.data import get_fnames
>>> f_name = get_fnames('aniso_vox')
>>> data, affine, zooms = load_nifti(f_name, return_voxsize=True)
>>> data.shape == (58, 58, 24)
True
>>> zooms
(4.0, 4.0, 5.0)
>>> new_zooms = (3.,3.,3.)
>>> new_zooms
(3.0, 3.0, 3.0)
>>> data2, affine2 = reslice(data, affine, zooms, new_zooms)
>>> data2.shape == (77, 77, 40)
True
```

1.2.4 scilpy.image.utils module

scilpy.image.utils.**check_slice_indices**(*vol_img*, *axis_name*, *slice_ids*)

Check that the given volume can be sliced at the given slice indices along the requested axis.

Parameters

- **vol_img** (*nib.Nifti1Image*) – Volume image.
- **axis_name** (*str*) – Slicing axis name.
- **slice_ids** (*array-like*) – Slice indices.

scilpy.image.utils.**count_non_zero_voxels**(*image*)

Count number of non zero voxels

image: string Path to the image

scilpy.image.utils.**extract_affine**(*input_files*)

Extract the affine from a list of nifti files.

Parameters **input_files** (*list of strings (file paths)*) – Diffusion data files.

Returns **affine** – Affine of the nifti volume.

Return type np.ndarray

scilpy.image.utils.**volume_iterator**(*img*, *blocksize=1*, *start=0*, *end=0*)

Generator that iterates on volumes of data.

Parameters

- **img** (*nib.Nifti1Image*) – Image of a 4D volume with shape X,Y,Z,N
- **blocksize** (*int, optional*) – Number of volumes to return in a single batch
- **start** (*int, optional*) – Starting iteration index in the 4D volume
- **end** (*int, optional*) – Stopping iteration index in the 4D volume (the volume at this index is excluded)

Yields *tuple of (list of int, ndarray)* – The ids of the selected volumes, and the selected data as a 4D array

1.3 scilpy.io package

1.3.1 scilpy.io.image module

scilpy.io.image.**assert_same_resolution**(*images*)
Check the resolution of multiple images. :param images: List of images or an image. :type images: list of string or string

scilpy.io.image.**get_data_as_mask**(*in_img*, *dtype=<class 'numpy.uint8'>*)
Get data as mask (force type np.uint8 or bool), check data type before casting.

Parameters

- **in_img** (*nibabel.nifti1.Nifti1Image*) – Image
- **dtype** (*data type for the output data (default: uint8)*) – type

Returns **data** – Data (dtype : np.uint8 or bool).

Return type numpy.ndarray

scilpy.io.image.**merge_labels_into_mask**(*atlas*, *filtering_args*)
Merge labels into a mask.

Parameters

- **atlas** (*np.ndarray*) – Atlas with labels as a numpy array (uint16) to merge.
- **filtering_args** (*str*) – Filtering arguments from the command line.

Returns **mask** – Mask obtained from the combination of multiple labels.

Return type nibabel.nifti1.Nifti1Image

1.3.2 scilpy.io.streamlines module

scilpy.io.streamlines.**check_tracts_same_format**(*parser*, *tractogram_1*, *tractogram_2*)
Assert that two filepaths have the same valid extension. :param parser: argparse.ArgumentParser object :param tractogram_1: Tractogram filename #1 :param tractogram_2: Tractogram filename #2

scilpy.io.streamlines.**ichunk**(*sequence*, *n*)
Yield successive n-sized chunks from sequence.

Parameters

- **sequence** (*numpy.ndarray*) – streamlines
- **n** (*int*) – amount of streamlines to load

Returns **chunck** – subset of streamlines

Return type list

scilpy.io.streamlines.**is_argument_set**(*args*, *arg_name*)

scilpy.io.streamlines.**load_tractogram_with_reference**(*parser*, *args*, *filepath*, *arg_name=None*)

Parameters

- **parser** (*Argument Parser*) – Used to print errors, if any.
- **args** (*Namespace*) – Parsed arguments. Used to get the 'ref' and 'bbox_check' args. See scilpy.io.utils to add the arguments to your parser.

- **filepath** (*str*) – Path of the tractogram file.
- **arg_name** (*str, optional*) – Name of the reference argument. By default the args.ref is used. If arg_name is given, then args.arg_name_ref will be used instead.

scilpy.io.streamlines.**reconstruct_streamlines**(*data*, *offsets*, *lengths*, *indices=None*)

Function to reconstruct streamlines from its data, offsets and lengths (from the nibabel tractogram object).

data [np.ndarray] Nx3 array representing all points of the streamlines.

offsets [np.ndarray] Nx1 array representing the cumsum of length array.

lengths [np.ndarray] Nx1 array representing the length of each streamline.

indices [list] List of int representing the indices to reconstruct.

Returns **streamlines** – List of streamlines.

Return type list of np.ndarray

scilpy.io.streamlines.**reconstruct_streamlines_from_hdf5**(*hdf5_filename*, *key=None*)

Function to reconstruct streamlines from hdf5, mainly to facilitate decomposition into thousand of connections and decrease I/O usage. ———- hdf5_filename : str

Filepath to the hdf5 file.

key [str] Key of the connection of interest (LABEL1_LABEL2).

Returns **streamlines** – List of streamlines.

Return type list of np.ndarray

scilpy.io.streamlines.**reconstruct_streamlines_from_memmap**(*memmap_filenames*, *indices=None*)

Function to reconstruct streamlines from memmaps, mainly to facilitate multiprocessing and decrease RAM usage.

memmap_filenames [tuple] Tuple of 3 filepath to numpy memmap (data, offsets, lengths).

indices [list] List of int representing the indices to reconstruct.

Returns **streamlines** – List of streamlines.

Return type list of np.ndarray

scilpy.io.streamlines.**streamlines_to_memmap**(*input_streamlines*)

Function to decompose on disk the array_sequence into its components. :param input_streamlines: All streamlines of the tractogram to segment. :type input_streamlines: ArraySequence

Returns **tmp_obj** – Temporary directory and tuple of filenames for the data, offsets and lengths.

Return type tuple

1.3.3 scilpy.io.utils module

scilpy.io.utils.**add_bbox_arg**(*parser*)

scilpy.io.utils.**add_force_b0_arg**(*parser*)

scilpy.io.utils.**add_json_args**(*parser*)

scilpy.io.utils.**add_overwrite_arg**(*parser*)

scilpy.io.utils.**add_processes_arg**(*parser*)

scilpy.io.utils.**add_reference_arg**(*parser*, *arg_name=None*)

scilpy.io.utils.**add_sh_basis_args**(*parser*, *mandatory=False*)
Add spherical harmonics (SH) bases argument.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser.
- **mandatory** (*bool*) – Whether this argument is mandatory.

scilpy.io.utils.**add_sphere_arg**(*parser*, *symmetric_only=False*, *default='symmetric724'*)

scilpy.io.utils.**add_verbose_arg**(*parser*)

scilpy.io.utils.**assert_fsl_options_exist**(*parser*, *options_args*, *command*)
Assert that all options for topup or eddy exist. If not, print parser's usage and exit.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser.
- **options_args** (*string*) – Options for fsl command
- **command** (*string*) – Command used (eddy or topup).

scilpy.io.utils.**assert_gradients_filenames_valid**(*parser*, *filename_list*, *gradient_format*)
Validate if gradients filenames follow BIDS or MRtrix convention

Parameters

- **parser** (*parser*) – Parser.
- **filename_list** (*list*) – list of gradient paths.
- **gradient_format** (*str*) – Can be either fsl or mrtrix.

scilpy.io.utils.**assert_inputs_exist**(*parser*, *required*, *optional=None*)
Assert that all inputs exist. If not, print parser's usage and exit.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser.
- **required** (*string or list of paths*) – Required paths to be checked.
- **optional** (*string or list of paths*) – Optional paths to be checked.

scilpy.io.utils.**assert_output_dirs_exist_and_empty**(*parser*, *args*, *required*, *optional=None*, *create_dir=True*)
Assert that all output directories exist. If not, print parser's usage and exit. If exists and not empty, and -f used, delete dirs.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser.
- **args** (*argparse namespace*) – Argument list.
- **required** (*string or list of paths to files*) – Required paths to be checked.
- **optional** (*string or list of paths to files*) – Optional paths to be checked.
- **create_dir** (*bool*) – If true, create the directory if it does not exist.

scilpy.io.utils.**assert_outputs_exist**(*parser*, *args*, *required*, *optional=None*, *check_dir_exists=True*)

Assert that all outputs don't exist or that if they exist, -f was used. If not, print parser's usage and exit.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser.
- **args** (*argparse namespace*) – Argument list.
- **required** (*string or list of paths to files*) – Required paths to be checked.
- **optional** (*string or list of paths to files*) – Optional paths to be checked.
- **check_dir_exists** (*bool*) – Test if output directory exists.

scilpy.io.utils.**check_tracts_same_format**(*parser*, *filename_list*)

scilpy.io.utils.**is_header_compatible_multiple_files**(*parser*, *list_files*, *verbose_all_compatible=False*)

Verifies the compatibility between the first item in list_files and the remaining files in list.

parser: argument parser Will raise an error if a file is not compatible.

list_files: List List of files to test

verbose_all_compatible: bool If true will print a message when everything is okay

scilpy.io.utils.**link_bundles_and_reference**(*parser*, *args*, *input_tractogram_list*)

Associate the bundle to their reference (if they require a reference). :param parser: Parser as created by argparse. :type parser: argparse.ArgumentParser object :param args: Args as created by argparse. :type args: argparse namespace :param input_tractogram_list: List of tractogram paths. :type input_tractogram_list: list

Returns list

Return type List of tuples, each matching one tractogram to a reference file.

scilpy.io.utils.**load_matrix_in_any_format**(*filepath*)

scilpy.io.utils.**parser_color_type**(*arg*)

Validate that a color component is between RBG values, else return an error From https://stackoverflow.com/a/55410582

scilpy.io.utils.**ranged_type**(*value_type*, *min_value*, *max_value*)

Return a function handle of an argument type function for ArgumentParser checking a range: *min_value* <= arg <= *max_value*.

Parameters

- **value_type** (*Type*) – Value-type to convert the argument.
- **min_value** (*scalar*) – Minimum acceptable argument value.
- **max_value** (*scalar*) – Maximum acceptable argument value.

Returns

- *Function handle of an argument type function for ArgumentParser.*
- *Usage*
- —— – ranged_type(float, 0.0, 1.0)

scilpy.io.utils.**read_info_from_mb_bdo**(*filename*)

scilpy.io.utils.**save_matrix_in_any_format**(*filepath*, *output_data*)

scilpy.io.utils.**snapshot**(*scene*, *filename*, ***kwargs*)
Wrapper around fury.window.snapshot For some reason, fury.window.snapshot flips the image vertically. This image unflips the image and then saves it.

scilpy.io.utils.**validate_nbr_processes**(*parser*, *args*)
Check if the passed number of processes arg is valid. Valid values are considered to be in the [0, CPU count] range:

- Raises a parser.error if an invalid value is provided.
- Returns the maximum number of cores retrieved if no value (or a value

of 0) is provided.

Parameters

- **parser** (*argparse.ArgumentParser object*) – Parser as created by argparse.
- **args** (*argparse namespace*) – Args as created by argparse.

Returns **nbr_cpu** – The number of CPU to be used.

Return type int

scilpy.io.utils.**validate_sh_basis_choice**(*sh_basis*)
Check if the passed sh_basis arg to a fct is right.

Parameters **sh_basis** (*str*) – Either 'descoteaux08' or 'tournier07'

Raises ValueError – If sh_basis is not one of 'descoteaux07' or 'tournier07'

scilpy.io.utils.**verify_compatibility_with_reference_sft**(*ref_sft*, *files_to_verify*, *parser*, *args*)
Verifies the compatibility of a reference sft with a list of files.

ref_sft: StatefulTractogram A tractogram to be used as reference.

files_to_verify: List[str] List of files that should be compatible with the reference sft. Files can be either other tractograms or nifti files (ex: masks).

parser: argument parser Will raise an error if a file is not compatible.

args: Namespace Should contain a args.reference if any file is a .tck, and possibly a args.bbox_check (set to True by default).

scilpy.io.utils.**verify_compression_th**(*compress_th*)
Verify that the compression threshold is between 0.001 and 1. Else, produce a warning.

Parameters **compress_th** (*float, the compression threshold.*) –

1.4 scilpy.preprocessing package

1.4.1 scilpy.preprocessing.distortion_correction module

scilpy.preprocessing.distortion_correction.**create_acqparams**(*readout*, *encoding_direction*, *nb_b0s=1*, *nb_rev_b0s=1*)
Create acqparams for Topup and Eddy

Parameters

- **readout** (*float*) – Readout time
- **encoding_direction** (*string*) – Encoding direction (x, y or z)
- **nb_b0s** (*int*) – Number of B=0 images
- **nb_rev_b0s** (*int*) – number of reverse b=0 images

Returns **acqparams** – acqparams

Return type np.array

scilpy.preprocessing.distortion_correction.**create_index**(*bvals*, *n_rev=0*)

Create index of bvals for Eddy

Parameters

- **bvals** (*np.array*) – b-values
- **n_rev** (*int, optional*) – Number of reverse phase images to take into account

Returns **index**

Return type np.array

scilpy.preprocessing.distortion_correction.**create_multi_topup_index**(*bvals*, *mean*, *n_rev*, *b0_thr=0*)

Create index of bvals for Eddy in cases where Topup ran on more than one b0 volume in both phase directions. The volumes must be ordered such as all forward phase acquisition are followed by all reverse phase ones (In the case of AP-PA, PA_1, PA_2, . . . , PA_N, AP_1, AP_2, . . . , AP_N).

Parameters

- **bvals** (*np.array*) – b-values
- **mean** (*string*) – Mean strategy used to subset the b0 volumes passed to topup (cluster or none)
- **n_rev** (*int, optional*) – Number of reverse phase images to take into account
- **b0_thr** (*int*) – All bvals under or equal to this threshold are considered as b0

Returns **index**

Return type np.array

scilpy.preprocessing.distortion_correction.**create_non_zero_norm_bvecs**(*bvecs*)

Add an epsilon to bvecs with a non zero norm. Mandatory for Topup and Eddy

Parameters **bvecs** (*np.array*) – b-vectors

Returns **bvecs** – b-vectors with an epsilon

Return type np.array

1.5 scilpy.reconst package

1.5.1 scilpy.reconst.afd_along_streamlines module

scilpy.reconst.afd_along_streamlines.**afd_and_rd_sums_along_streamlines**(*sft*, *fodf*, *fodf_basis*, *length_weighting*)

Compute the mean Apparent Fiber Density (AFD) and mean Radial fODF (radfODF) maps along a bundle.

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines needed.
- **fodf** (*nibabel.image*) – fODF with shape (X, Y, Z, #coeffs). #coeffs depend on the sh_order.
- **fodf_basis** (*string*) – Has to be descoteaux07 or tournier07.
- **length_weighting** (*bool*) – If set, will weigh the AFD values according to segment lengths.

Returns

- **afd_sum_map** (*np.array*) – AFD map.
- **rd_sum_map** (*np.array*) – fdAFD map.
- **weight_map** (*np.array*) – Segment lengths.

scilpy.reconst.afd_along_streamlines.**afd_map_along_streamlines**(*sft*, *fodf*, *fodf_basis*, *length_weighting*)

Compute the mean Apparent Fiber Density (AFD) and mean Radial fODF (radfODF) maps along a bundle.

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines needed.
- **fodf** (*nibabel.image*) – fODF with shape (X, Y, Z, #coeffs) coeffs depending on the sh_order
- **fodf_basis** (*string*) – Has to be descoteaux07 or tournier07
- **length_weighting** (*bool*) – If set, will weigh the AFD values according to segment lengths

Returns

- **afd_sum** (*np.array*) – AFD map (weighted if length_weighting)
- **rd_sum** (*np.array*) – rdAFD map (weighted if length_weighting)

1.5.2 scilpy.reconst.fodf module

scilpy.reconst.fodf.**compute_fodf**(*data*, *bvals*, *bvecs*, *full_frf*, *sh_order=8*, *nbr_processes=None*, *mask=None*, *sh_basis='descoteaux07'*, *return_sh=True*, *n_peaks=5*, *force_b0_threshold=False*)

Script to compute Constrained Spherical Deconvolution (CSD) fiber ODFs.

By default, will output all possible files, using default names. Specific names can be specified using the file flags specified in the "File flags" section.

If –not_all is set, only the files specified explicitly by the flags will be output.

See [Tournier et al. NeuroImage 2007] and [Cote et al Tractometer MedIA 2013] for quantitative comparisons with Sharpening Deconvolution Transform (SDT).

Parameters

- **data** (*ndarray*) – 4D Input diffusion volume with shape (X, Y, Z, N)
- **bvals** (*ndarray*) – 1D bvals array with shape (N,)
- **bvecs** (*ndarray*) – 2D (normalized) bvecs array with shape (N, 3)
- **full_frf** (*ndarray*) – frf data, ex, loaded from a frf_file, with shape (4,).
- **sh_order** (*int, optional*) – SH order used for the CSD. (Default: 8)
- **nbr_processes** (*int, optional*) – Number of sub processes to start. Default = none, i.e use the cpu count. If 0, use all processes.
- **mask** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary mask. Only the data inside the mask will be used for computations and reconstruction. Useful if no white matter mask is available.
- **sh_basis** (*str, optional*) – Spherical harmonics basis used for the SH coefficients.Must be either 'descoteaux07' or 'tournier07' (default 'descoteaux07') - 'descoteaux07': SH basis from the Descoteaux et al. MRM 2007 paper - 'tournier07': SH basis from the Tournier et al. NeuroImage 2007 paper.
- **return_sh** (*bool, optional*) – If true, returns the sh.
- **n_peaks** (*int, optional*) – Nb of peaks for the fodf. Default: copied dipy's default, i.e. 5.
- **force_b0_threshold** (*bool, optional*) – If True, will continue even if the minimum bvalue is suspiciously high.

Returns peaks_csd – An object with `gfa`, `peak_directions`, `peak_values`, `peak_indices`, `odf`, `shm_coeffs` as attributes

Return type PeaksAndMetrics

1.5.3 scilpy.reconst.frf module

scilpy.reconst.frf.**compute_msmt_frf**(*data*, *bvals*, *bvecs*, *btens=None*, *data_dti=None*, *bvals_dti=None*, *bvecs_dti=None*, *btens_dti=None*, *mask=None*, *mask_wm=None*, *mask_gm=None*, *mask_csf=None*, *fa_thr_wm=0.7*, *fa_thr_gm=0.2*, *fa_thr_csf=0.1*, *md_thr_gm=0.0007*, *md_thr_csf=0.003*, *min_nvox=300*, *roi_radii=10*, *roi_center=None*, *tol=20*, *force_b0_threshold=False*)

Compute a multi-shell, multi-tissue single Fiber Response Function from a DWI volume. A DTI fit is made, and voxels containing a single fiber population are found using a threshold on the FA and MD.

Parameters

- **data** (*ndarray*) – 4D Input diffusion volume with shape (X, Y, Z, N)
- **bvals** (*ndarray*) – 1D bvals array with shape (N,)
- **bvecs** (*ndarray*) – 2D bvecs array with shape (N, 3)

- **mask** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary mask. Only the data inside the mask will be used for computations and reconstruction.
- **mask_wm** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary white matter mask. Only the data inside this mask will be used to estimate the fiber response function of WM.
- **mask_gm** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary grey matter mask. Only the data inside this mask will be used to estimate the fiber response function of GM.
- **mask_csf** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary csf mask. Only the data inside this mask will be used to estimate the fiber response function of CSF.
- **fa_thr_wm** (*float, optional*) – Use this threshold to select single WM fiber voxels from the FA inside the WM mask defined by mask_wm. Each voxel above this threshold will be selected. Defaults to 0.7
- **fa_thr_gm** (*float, optional*) – Use this threshold to select GM voxels from the FA inside the GM mask defined by mask_gm. Each voxel below this threshold will be selected. Defaults to 0.2
- **fa_thr_csf** (*float, optional*) – Use this threshold to select CSF voxels from the FA inside the CSF mask defined by mask_csf. Each voxel below this threshold will be selected. Defaults to 0.1
- **md_thr_gm** (*float, optional*) – Use this threshold to select GM voxels from the MD inside the GM mask defined by mask_gm. Each voxel below this threshold will be selected. Defaults to 0.0007
- **md_thr_csf** (*float, optional*) – Use this threshold to select CSF voxels from the MD inside the CSF mask defined by mask_csf. Each voxel below this threshold will be selected. Defaults to 0.003
- **min_nvox** (*int, optional*) – Minimal number of voxels needing to be identified as single fiber voxels in the automatic estimation. Defaults to 300.
- **roi_radii** (*int or array-like (3,), optional*) – Use those radii to select a cuboid roi to estimate the FRF. The roi will be a cuboid spanning from the middle of the volume in each direction with the different radii. Defaults to 10.
- **roi_center** (*tuple(3), optional*) – Use this center to span the roi of size roi_radius (center of the 3D volume).
- **tol** (*int*) – tolerance gap for b-values clustering. Defaults to 20
- **force_b0_threshold** (*bool, optional*) – If set, will continue even if the minimum bvalue is suspiciously high.

Returns

- **reponses** (*list of ndarray*) – Fiber Response Function of each (3) tissue type, with shape (4, N).
- **frf_masks** (*list of ndarray*) – Mask where the frf was calculated, for each (3) tissue type, with shape (X, Y, Z).

Raises `ValueError` – If less than *min_nvox* voxels were found with sufficient FA to estimate the FRF.

scilpy.reconst.frf.**compute_ssst_frf**(*data*, *bvals*, *bvecs*, *mask=None*, *mask_wm=None*, *fa_thresh=0.7*, *min_fa_thresh=0.5*, *min_nvox=300*, *roi_radii=10*, *roi_center=None*, *force_b0_threshold=False*)

Compute a single-shell (under b=1500), single-tissue single Fiber Response Function from a DWI volume. A DTI fit is made, and voxels containing a single fiber population are found using a threshold on the FA.

Parameters

- **data** (*ndarray*) – 4D Input diffusion volume with shape (X, Y, Z, N)
- **bvals** (*ndarray*) – 1D bvals array with shape (N,)
- **bvecs** (*ndarray*) – 2D bvecs array with shape (N, 3)
- **mask** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary mask. Only the data inside the mask will be used for computations and reconstruction. Useful if no white matter mask is available.
- **mask_wm** (*ndarray, optional*) – 3D mask with shape (X,Y,Z) Binary white matter mask. Only the data inside this mask and above the threshold defined by fa_thresh will be used to estimate the fiber response function.
- **fa_thresh** (*float, optional*) – Use this threshold as the initial threshold to select single fiber voxels. Defaults to 0.7
- **min_fa_thresh** (*float, optional*) – Minimal value that will be tried when looking for single fiber voxels. Defaults to 0.5
- **min_nvox** (*int, optional*) – Minimal number of voxels needing to be identified as single fiber voxels in the automatic estimation. Defaults to 300.
- **roi_radii** (*int or array-like (3,), optional*) – Use those radii to select a cuboid roi to estimate the FRF. The roi will be a cuboid spanning from the middle of the volume in each direction with the different radii. Defaults to 10.
- **roi_center** (*tuple(3), optional*) – Use this center to span the roi of size roi_radius (center of the 3D volume).
- **force_b0_threshold** (*bool, optional*) – If set, will continue even if the minimum bvalue is suspiciously high.

Returns **full_reponse** – Fiber Response Function, with shape (4,)

Return type ndarray

Raises `ValueError` – If less than *min_nvox* voxels were found with sufficient FA to estimate the FRF.

1.5.4 scilpy.reconst.raw_signal module

scilpy.reconst.raw_signal.**compute_dwi_attenuation**(*dwi_weights: numpy.ndarray*, *b0: numpy.ndarray*)

Compute signal attenuation by dividing the dwi signal with the b0.

dwi_weights [np.ndarray of shape (X, Y, Z, #gradients)] Diffusion weighted images.

b0 [np.ndarray of shape (X, Y, Z)] B0 image.

Returns **dwi_attenuation** – Signal attenuation (Diffusion weights normalized by the B0).

Return type np.ndarray

scilpy.reconst.raw_signal.**compute_sh_coefficients**(*dwi*, *gradient_table*, *sh_order=4*, *basis_type='descoteaux07'*, *smooth=0.006*, *use_attenuation=False*, *force_b0_threshold=False*, *mask=None*, *sphere=None*)

Fit a diffusion signal with spherical harmonics coefficients.

Parameters

- **dwi** (*nib.Nifti1Image object*) – Diffusion signal as weighted images (4D).
- **gradient_table** (*GradientTable*) – Dipy object that contains all bvals and bvecs.
- **sh_order** (*int, optional*) – SH order to fit, by default 4.
- **smooth** (*float, optional*) – Lambda-regularization coefficient in the SH fit, by default 0.006.
- **basis_type** (*str*) – Either 'tournier07' or 'descoteaux07'
- **use_attenuation** (*bool, optional*) – If true, we will use DWI attenuation. [False]
- **force_b0_threshold** (*bool, optional*) – If set, will continue even if the minimum bvalue is suspiciously high.
- **mask** (*nib.Nifti1Image object, optional*) – Binary mask. Only data inside the mask will be used for computations and reconstruction.
- **sphere** (*Sphere*) – Dipy object. If not provided, will use Sphere(xyz=bvecs).

Returns **sh_coeffs** – Spherical harmonics coefficients at every voxel. The actual number of coefficients depends on *sh_order*.

Return type np.ndarray with shape (X, Y, Z, #coeffs)

1.5.5 scilpy.reconst.utils module

scilpy.reconst.utils.**find_order_from_nb_coeff**(*data*)

scilpy.reconst.utils.**get_b_matrix**(*order*, *sphere*, *sh_basis_type*, *return_all=False*)

scilpy.reconst.utils.**get_maximas**(*data*, *sphere*, *b_matrix*, *threshold*, *absolute_threshold*, *min_separation_angle=25*)

scilpy.reconst.utils.**get_sh_order_and_fullness**(*ncoeffs*)

Get the order of the SH basis from the number of SH coefficients as well as a boolean indicating if the basis is full.

scilpy.reconst.utils.**get_sphere_neighbours**(*sphere*, *max_angle*)

Get a matrix of neighbours for each direction on the sphere, within the min_separation_angle.

min_separation_angle: float Maximum angle in radians defining the neighbourhood of each direction.

Returns **neighbours** – Neighbour directions for each direction on the sphere.

Return type ndarray

1.6 scilpy.segment package

1.6.1 scilpy.segment.models module

scilpy.segment.models.**remove_similar_streamlines**(*streamlines*, *threshold=5*)

Remove similar streamlines, shuffling streamlines will impact the results. Only provide a small set of streamlines (below 2000 if possible).

Parameters

- **streamlines** (*list of numpy.ndarray*) – Input streamlines to remove duplicates from.
- **threshold** (*float*) – Distance threshold to consider two streamlines similar, in mm.

Returns **streamlines**

Return type list of numpy.ndarray

1.6.2 scilpy.segment.recobundlesx module

class scilpy.segment.recobundlesx.**RecobundlesX**(*memmap_filenames*, *wb_clusters_indices*, *wb_centroids*, *nb_points=20*, *slr_num_thread=1*, *rng=None*)

Bases: object

This class is a 'remastered' version of the Dipy Recobundles class. Made to be used in sync with the voting_scheme. Adapted in many way for HPC processing and increase control over parameters and logging. However, in essence they do the same thing. https://github.com/nipy/dipy/blob/master/dipy/segment/bundles.py

References

get_final_pruned_indices()

Public getter for the final indices recognize by the algorithm.

prune_far_from_model(*bundle_pruning_thr=10*, *neighbors_cluster_thr=8*)

Wrapper function to prune clusters from the tractogram too far from the model. :param neighbors_to_prune, list or arraySequence, streamlines to prune. :param bundle_pruning_thr, float, distance in mm for pruning. :param neighbors_cluster_thr, float, distance in mm for clustering.

recognize(*model_bundle*, *model_clust_thr=8*, *bundle_pruning_thr=8*, *slr_transform_type='similarity'*, *identifier=None*)

Parameters

- **model_bundle** (*list or ArraySequence*) – Model bundle as loaded by the nibabel API.
- **model_clust_thr** (*obj*) – Distance threshold (mm) for model clustering (QBx)
- **bundle_pruning_thr** (*int*) – Distance threshold (mm) for the final pruning.
- **slr_transform_type** (*str*) – Define the transformation for the local SLR. [translation, rigid, similarity, scaling]
- **identifier** (*str*) – Identify the current bundle being recognized for the logging.

Returns **clusters** – Streamlines that were recognized by Recobundles and these parameters.

Return type list

1.6.3 scilpy.segment.streamlines module

scilpy.segment.streamlines.**filter_cuboid**(*sft*, *cuboid_radius*, *cuboid_center*, *filter_type*, *is_exclude*)

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **cuboid_radius** (*numpy.ndarray (3)*) – Size in mm, x/y/z of the cuboid.
- **cuboid_center** (*numpy.ndarray (3)*) – Center x/y/z of the cuboid.
- **filter_type** (*str*) – One of the 3 following choices, 'any', 'all', 'either_end', 'both_ends'.
- **is_exclude** (*bool*) – Value to indicate if the ROI is an AND (false) or a NOT (true).
- **is_in_vox** (*bool*) – Value to indicate if the ROI is in voxel space.

Returns **ids** – Filtered sft. Ids of the streamlines passing through the mask.

Return type tuple

scilpy.segment.streamlines.**filter_ellipsoid**(*sft*, *ellipsoid_radius*, *ellipsoid_center*, *filter_type*, *is_exclude*, *is_in_vox=False*)

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **ellipsoid_radius** (*numpy.ndarray (3)*) – Size in mm, x/y/z of the ellipsoid.
- **ellipsoid_center** (*numpy.ndarray (3)*) – Center x/y/z of the ellipsoid.
- **filter_type** (*str*) – One of the 3 following choices, 'any', 'all', 'either_end', 'both_ends'.
- **is_exclude** (*bool*) – Value to indicate if the ROI is an AND (false) or a NOT (true).
- **is_in_vox** (*bool*) – Value to indicate if the ROI is in voxel space.

Returns **ids** – Filtered sft. Ids of the streamlines passing through the mask.

Return type tuple

scilpy.segment.streamlines.**filter_grid_roi**(*sft*, *mask*, *filter_type*, *is_exclude*)

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **mask** (*numpy.ndarray*) – Binary mask in which the streamlines should pass.
- **filter_type** (*str*) – One of the 3 following choices, 'any', 'all', 'either_end', 'both_ends'.
- **is_exclude** (*bool*) – Value to indicate if the ROI is an AND (false) or a NOT (true).

Returns

- **new_sft** (*StatefulTractogram*) – Filtered sft.

- **ids** (*list*) – Ids of the streamlines passing through the mask.

scilpy.segment.streamlines.**filter_grid_roi_both**(*sft*, *mask_1*, *mask_2*)

Filters streamlines with one end in a mask and the other in another mask.

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **mask_1** (*numpy.ndarray*) – Binary mask in which the streamlines should start or end.
- **mask_2** (*numpy.ndarray*) – Binary mask in which the streamlines should start or end.

Returns

- **new_sft** (*StatefulTractogram*) – Filtered sft.
- **ids** (*list*) – Ids of the streamlines passing through the mask.

scilpy.segment.streamlines.**pre_filtering_for_geometrical_shape**(*sft*, *size*, *center*, *filter_type*, *is_in_vox*)

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **size** (*numpy.ndarray (3)*) – Size in mm, x/y/z of the ROI.
- **center** (*numpy.ndarray (3)*) – Center x/y/z of the ROI.
- **filter_type** (*str*) – One of the 3 following choices, 'any', 'all', 'either_end', 'both_ends'.
- **is_in_vox** (*bool*) – Value to indicate if the ROI is in voxel space.

Returns **ids** – Filtered sft. Ids of the streamlines passing through the mask.

Return type tuple

scilpy.segment.streamlines.**streamlines_in_mask**(*sft*, *target_mask*, *all_in=False*)

Parameters

- **sft** (*StatefulTractogram*) – StatefulTractogram containing the streamlines to segment.
- **target_mask** (*numpy.ndarray*) – Binary mask in which the streamlines should pass.

Returns **ids** – Ids of the streamlines passing through the mask.

Return type list

1.6.4 scilpy.segment.voting_scheme module

class scilpy.segment.voting_scheme.**VotingScheme**(*config*, *atlas_directory*, *transformation*, *output_directory*, *tractogram_clustering_thr*, *nb_points=12*, *minimal_vote_ratio=0.5*, *multi_parameters=1*)

Bases: object

scilpy.segment.voting_scheme.**single_clusterize_and_rbx_init**(*args*)
Wrapper function to multiprocess clustering executions and recobundles initialisation.

Parameters

- **tmp_memmap_filename** (*tuple (3)*) – Temporary filename for the data, offsets and lengths.
- **parameters_list** (*tuple (3)*) –

  clustering_thr [int] Distance in mm (for QBx) to cluster the input tractogram.

  seed [int] Value to initialize the RandomState of numpy.

  nb_points [int] Number of points used for all resampling of streamlines.

Returns **rbx** – Initialisation of the recobundles class using specific parameters.

Return type dict

scilpy.segment.voting_scheme.**single_recognize**(*args*)
Wrapper function to multiprocess recobundles execution.

Parameters

- **rbx_dict** (*dict*) – Dictionary with int as key and QBx ClusterMap as values
- **model_dict** (*dict*) – Dictionary with model tag as key and model streamlines as values
- **parameters_list** (*tuple (8)*) –

  bundle_id [int] Unique value to each bundle to identify them

  tag [str] Model bundle filepath for logging

  tct [int] Tractogram clustering threshold, distance in mm (for QBx)

  mct [int] Model clustering threshold, distance in mm (for QBx)

  bpt [int] Bundle pruning threshold, distance in mm

  slr_transform_type [str] Define the transformation for the local SLR [translation, rigid, similarity, scaling]

  seed [int] Value to initialize the RandomState of numpy

Returns

transf_neighbor –

bundle_id [(int)] Unique value to each bundle to identify them.

recognized_indices [(numpy.ndarray)] Streamlines indices from the original tractogram.

Return type tuple

1.7 scilpy.tracking package

1.7.1 scilpy.tracking.tools module

scilpy.tracking.tools.**filter_streamlines_by_length**(*sft*, *min_length=0.0*, *max_length=inf*)
Filter streamlines using minimum and max length.

Parameters

- **sft** (*StatefulTractogram*) – SFT containing the streamlines to filter.
- **min_length** (*float*) – Minimum length of streamlines, in mm.
- **max_length** (*float*) – Maximum length of streamlines, in mm.

Returns **filtered_sft** – A tractogram without short streamlines.

Return type StatefulTractogram

scilpy.tracking.tools.**filter_streamlines_by_total_length_per_dim**(*sft*, *limits_x*, *limits_y*, *limits_z*, *use_abs*, *save_rejected*)

Filter streamlines using sum of abs length per dimension.

Note: we consider that x, y, z are the coordinates of the streamlines; we do not verify if they are aligned with the brain's orientation.

Parameters

- **sft** (*StatefulTractogram*) – SFT containing the streamlines to filter.
- **limits_x** (*[float float]*) – The list of [min, max] for the x coordinates.
- **limits_y** (*[float float]*) – The list of [min, max] for the y coordinates.
- **limits_z** (*[float float]*) – The list of [min, max] for the z coordinates.
- **use_abs** (*bool*) – If True, will use the total of distances in absolute value (ex, coming back on yourself will contribute to the total distance instead of cancelling it).
- **save_rejected** (*bool*) – If true, also returns the SFT of rejected streamlines. Else, returns None.

Returns

- **filtered_sft** (*StatefulTractogram*) – A tractogram of accepted streamlines.
- **ids** (*list*) – The list of good ids.
- **rejected_sft** (*StatefulTractogram or None*) – A tractogram of rejected streamlines.

scilpy.tracking.tools.**get_theta**(*requested_theta*, *tracking_type*)

scilpy.tracking.tools.**resample_streamlines_num_points**(*sft*, *num_points*)

Resample streamlines using number of points per streamline

Parameters

- **sft** (*StatefulTractogram*) – SFT containing the streamlines to subsample.
- **num_points** (*int*) – Number of points per streamline in the output.

Returns **resampled_sft** – The resampled streamlines as a sft.

Return type StatefulTractogram

scilpy.tracking.tools.**resample_streamlines_step_size**(*sft*, *step_size*)

Resample streamlines using a fixed step size.

Parameters

- **sft** (*StatefulTractogram*) – SFT containing the streamlines to subsample.
- **step_size** (*float*) – Size of the new steps, in mm.

Returns **resampled_sft** – The resampled streamlines as a sft.

Return type StatefulTractogram

scilpy.tracking.tools.**sample_distribution**(*dist*)

Parameters **dist** (*numpy.array*) – The empirical distribution to sample from.

Returns **ind** – The index of the sampled element.

Return type int

scilpy.tracking.tools.**smooth_line_gaussian**(*streamline*, *sigma*)

scilpy.tracking.tools.**smooth_line_spline**(*streamline*, *sigma*, *nb_ctrl_points*)

1.8 scilpy.tractanalysis package

1.8.1 scilpy.tractanalysis.distance_to_centroid module

scilpy.tractanalysis.distance_to_centroid.**min_dist_to_centroid**(*bundle_pts*, *centroid_pts*)

1.8.2 scilpy.tractanalysis.features module

scilpy.tractanalysis.features.**detect_ushape**(*sft*, *minU*, *maxU*)
Extract streamlines depending of their "u-shapeness". :param sft: Tractogram used to extract streamlines depending on their ushapeness. :type sft: Statefull tractogram :param minU: Minimum ufactor of a streamline. :type minU: Float :param maxU: Maximum ufactor of a streamline. :type maxU: Float

Returns **list** – Only the ids are returned so proper filtering can be done afterwards.

Return type the ids of clean streamlines

scilpy.tractanalysis.features.**get_streamlines_bounding_box**(*streamlines*)
Classify inliers and outliers from a list of streamlines. :param streamlines: The list of streamlines from which inliers and outliers are separated. :type streamlines: list of ndarray

Returns **tuple** – bounding box

Return type Minimum and maximum corner coordinate of the streamlines

scilpy.tractanalysis.features.**get_streamlines_centroid**(*streamlines*, *nb_points*)
Compute centroid from streamlines using QuickBundles.

Parameters

- **streamlines** (*list of ndarray*) – The list of streamlines from which we compute the centroid.
- **nb_points** (*int*) – Number of points defining the centroid streamline.

Returns

Return type List of length one, containing a np.ndarray of shape (nb_points, 3)

scilpy.tractanalysis.features.**outliers_removal_using_hierarchical_quickbundles**(*streamlines*, *nb_points=12*, *min_threshold=0*, *nb_samplings_m*, *sampling_seed=1234*, *fast_approx=Fal*

Classify inliers and outliers from a list of streamlines. :param streamlines: The list of streamlines from which inliers and outliers are separated. :type streamlines: list of ndarray :param min_threshold: Quickbundles distance threshold for the last threshold. :type min_threshold: float :param nb_samplings_max: Number of run executed to explore the search space.

A different sampling is used each time.

Parameters **sampling_seed** (*int*) – Random number generation initialization seed.

Returns **ndarray**

Return type Float value representing the 0-1 score for each streamline

scilpy.tractanalysis.features.**prune**(*streamlines*, *threshold*, *features*)

Discriminate streamlines based on a metrics, usually summary from function outliers_removal_using_hierarchical_quickbundles. :param streamlines: The list of streamlines from which inliers and outliers are separated. :type streamlines: list of ndarray :param threshold: Threshold use to discriminate streamlines using the feature. :type threshold: float :param features: Values that represent a relevant metric to disciminate streamlines. :type features: ndarray

Returns Indices for outliers (below threshold), indices for inliers (above threshold).

Return type tuple

scilpy.tractanalysis.features.**remove_loops_and_sharp_turns**(*streamlines*, *max_angle*, *use_qb=False*, *qb_threshold=15.0*, *qb_seed=0*, *num_processes=1*)

Remove loops and sharp turns from a list of streamlines. :param streamlines: The list of streamlines from which to remove loops and sharp turns. :type streamlines: list of ndarray :param max_angle: Maximal winding angle a streamline can have before

being classified as a loop.

Parameters

- **use_qb** (*bool*) – Set to True if the additional QuickBundles pass is done. This will help remove sharp turns. Should only be used on bundled streamlines, not on whole-brain tractograms.
- **qb_threshold** (*float*) – Quickbundles distance threshold, only used if use_qb is True.
- **qb_seed** (*int*) – Seed to initialize randomness in QuickBundles
- **num_processes** (*int*) – Split the calculation to a pool of children processes.

Returns **list** – Only the ids are returned so proper filtering can be done afterwards

Return type the ids of clean streamlines

scilpy.tractanalysis.features.**remove_outliers**(*streamlines*, *threshold*, *nb_points=12*, *nb_samplings=30*, *fast_approx=False*)

Wrapper to classify inliers and outliers from a list of streamlines. :param streamlines: The list of streamlines from which inliers and outliers are separated. :type streamlines: list of ndarray :param threshold: Quickbundles distance threshold for the last threshold. :type threshold: float :param ——-: :param A tuple containing: list: streamlines considered inliers

list: streamlines considered outliers

1.8.3 scilpy.tractanalysis.grid_intersections module

scilpy.tractanalysis.grid_intersections.**grid_intersections**()

1.8.4 scilpy.tractanalysis.grid_intersections module

scilpy.tractanalysis.grid_intersections.**grid_intersections**()

1.8.5 scilpy.tractanalysis.quick_tools module

scilpy.tractanalysis.quick_tools.**get_next_real_point**()

scilpy.tractanalysis.quick_tools.**get_previous_real_point**()

1.8.6 scilpy.tractanalysis.quick_tools module

scilpy.tractanalysis.quick_tools.**get_next_real_point**()

scilpy.tractanalysis.quick_tools.**get_previous_real_point**()

1.8.7 scilpy.tractanalysis.reproducibility_measures module

scilpy.tractanalysis.reproducibility_measures.**approximate_surface_node**(*roi*)

Compute the number of surface voxels (i.e. nodes connected to at least one zero-valued neighboring voxel) :param roi: A ndarray where voxel values represent the density of a bundle and

it is binarized.

Returns **int**

Return type the number of surface voxels

scilpy.tractanalysis.reproducibility_measures.**binary_classification**(*segmentation_indices*, *gold_standard_indices*, *original_count*, *mask_count=0*)

Compute all the binary classification measures using only indices from a dataset and its ground truth in any representation (voxels or streamlines). ———- segmentation_indices: list of int

Indices of the data that are part of the segmentation.

gold_standard_indices: list of int Indices of the ground truth.

original_count: int Total size of the original dataset (before segmentation), e.g len(streamlines) or np.prod(mask.shape).

mask_count: int Number of non-zeros voxels in the original dataset. Needed in order to get a valid true positive count for the voxel representation.

Returns

float: Value between 0 and 1 that represent the spatial aggrement between both bundles.

list of ndarray: intersection_robust of streamlines in both bundle list of ndarray: union_robust of streamlines in both bundle

Return type A tuple containing

scilpy.tractanalysis.reproducibility_measures.**compute_bundle_adjacency_streamlines**(*bundle_1*, *bundle_2*, *non_overla* *centroids_1=N* *centroids_2=N*

Compute the distance in millimeters between two bundles. Uses centroids to limit computation time. Each centroid of the first bundle is matched to the nearest centroid of the second bundle and vice-versa. Distance between matched paired is averaged for the final results. .. rubric:: References

Parameters

- **bundle_1** (*list of ndarray*) – First set of streamlines.
- **bundle_2** (*list of ndarray*) – Second set of streamlines.
- **non_overlap** (*bool*) – Exclude overlapping streamlines from the computation.
- **centroids_1** (*list of ndarray*) – Pre-computed centroids for the first bundle.
- **centroids_2** (*list of ndarray*) – Pre-computed centroids for the second bundle.

Returns **float**

Return type Distance in millimeters between both bundles.

scilpy.tractanalysis.reproducibility_measures.**compute_bundle_adjacency_voxel**(*binary_1*, *binary_2*, *non_overlap=False*

Compute the distance in millimeters between two bundles in the voxel representation. Convert the bundles to binary masks. Each voxel of the first bundle is matched to the the nearest voxel of the second bundle and vice-versa. Distance between matched paired is averaged for the final results. :param binary_1: Binary mask computed from the first bundle :type binary_1: ndarray :param binary_2: Binary mask computed from the second bundle :type binary_2: ndarray :param non_overlap: Exclude overlapping voxels from the computation. :type non_overlap: bool

Returns **float**

Return type Distance in millimeters between both bundles.

scilpy.tractanalysis.reproducibility_measures.**compute_correlation**(*density_1*, *density_2*)

Compute the overlap (dice coefficient) between two density maps (or binary). Correlation being less robust to extreme case (no overlap, identical array), a lot of check a needed to prevent NaN. :param density_1: Density (or binary) map computed from the first bundle :type density_1: ndarray :param density_2: Density (or binary) map computed from the second bundle :type density_2: ndarray

Returns **float** – between both bundles taking into account density.

Return type Value between 0 and 1 that represent the spatial aggrement

scilpy.tractanalysis.reproducibility_measures.**compute_dice_streamlines**(*bundle_1*, *bundle_2*)

Compute the overlap (dice coefficient) between two bundles. Both bundles need to come from the exact same

tractogram. :param bundle_1: First set of streamlines. :type bundle_1: list of ndarray :param bundle_2: Second set of streamlines. :type bundle_2: list of ndarray

Returns

float: Value between 0 and 1 that represent the spatial aggrement between both bundles.

list of ndarray: intersection_robust of streamlines in both bundle list of ndarray: union_robust of streamlines in both bundle

Return type A tuple containing

scilpy.tractanalysis.reproducibility_measures.**compute_dice_voxel**(*density_1*, *density_2*)

Compute the overlap (dice coefficient) between two density maps (or binary). :param density_1: Density (or binary) map computed from the first bundle :type density_1: ndarray :param density_2: Density (or binary) map computed from the second bundle :type density_2: ndarray

Returns

float: Value between 0 and 1 that represent the spatial aggrement between both bundles.

float: Value between 0 and 1 that represent the spatial aggrement between both bundles, weighted by streamlines density.

Return type A tuple containing

scilpy.tractanalysis.reproducibility_measures.**compute_fractal_dimension**(*density*, *n_steps=10*, *box_size_min=1.0*, *box_size_max=2.0*, *threshold=0.0*, *box_size=None*)

Compute the fractal dimension of a bundle to measure the roughness. The code is extracted from https://github.com/FBK-NILab/fractal_dimension Parameters. The result is dependent on voxel size and the number of voxels. If data comparison is performed, the bundles MUST be in same resolution. ———- density: ndarray

A ndarray where voxel values represent the density of a bundle. This function computes the fractal dimension of the bundle.

n_steps: int The number of box sizes used to approximate fractal dimension. A larger number of steps will increase the accuracy of the approximation, but will also take more time. The default number of boxes sizes is 10.

box_size_min: float The minimum size of boxes. This number should be larger than or equal to 1.0 and is defaulted to 1.0.

box_size_max: float The maximum size of boxes. This number should be larger than the minimum size of boxes.

threshold: float The threshold to filter the voxels in the density array. The default is set to 0, so only nonzero voxels will be considered.

box_size: ndarray A ndarray of different sizes of boxes in a linear space in an ascending order.

Returns float

Return type fractal dimension of a bundle

scilpy.tractanalysis.reproducibility_measures.**get_endpoints_density_map**(*streamlines*, *dimensions*, *point_to_select=1*)

Compute an endpoints density map, supports selecting more than one points at each end. :param streamlines: The list of streamlines to compute endpoints density from. :type streamlines: list of ndarray :param dimensions: The shape of the reference volume for the streamlines. :type dimensions: tuple :param point_to_select: Instead of computing the density based on the first and last points,

select more than one at each end. To support compressed streamlines, a resampling to 0.5mm per segment is performed.

Returns **ndarray**

Return type A ndarray where voxel values represent the density of endpoints.

scilpy.tractanalysis.reproducibility_measures.**get_head_tail_density_maps**(*streamlines*, *dimensions*, *point_to_select=1*)

Compute two separate endpoints density maps for the head and tail :param streamlines: The list of streamlines to compute endpoints density from. :type streamlines: list of ndarray :param dimensions: The shape of the reference volume for the streamlines. :type dimensions: tuple :param point_to_select: Instead of computing the density based on the first and last points,

select more than one at each end. To support compressed streamlines, a resampling to 0.5mm per segment is performed.

Returns

ndarray: A ndarray where voxel values represent the density of head endpoints.

ndarray: A ndarray where voxel values represent the density of tail endpoints.

Return type A tuple containing

1.8.8 scilpy.tractanalysis.streamlines_metrics module

scilpy.tractanalysis.streamlines_metrics.**compute_tract_counts_map**()

1.8.9 scilpy.tractanalysis.streamlines_metrics module

scilpy.tractanalysis.streamlines_metrics.**compute_tract_counts_map**()

1.8.10 scilpy.tractanalysis.todi module

class scilpy.tractanalysis.todi.**TrackOrientationDensityImaging**(*img_shape*, *sphere_type='repulsion724'*)

Bases: object

compute_average_dir()

Voxel-wise average of TODI orientations.

Average all orientation of each voxel, of the TODI map, into a single direction, warning for crossing.

Returns **avg_dir** – Volume containing a single 3-vector (peak) per voxel.

Return type numpy.ndarray (4D)

compute_distance_to_peak(*peak_img*, *normalize_count=True*, *deg=True*, *with_avg_dir=True*)
Compute distance to peak map.

Compute the distance of the TODI map to peaks at each position, in radian or degree.

Parameters

- **peak_img** (*numpy.ndarray (4D)*) – Peaks image, as written by most of Scilpy scripts.
- **normalize_count** (*bool, optional*) – Normalize/weight the error map by the density map (default True).
- **deg** (*bool, optional*) – Returned error map as degree instead of radian (default True).
- **with_avg_dir** (*bool, optional*) – Average all orientation of each voxel of the TODI map into a single direction, warning for crossing (default True).

Returns **error_map** – Average angle error map per voxel.

Return type numpy.ndarray (3D)

compute_todi(*streamlines*, *length_weights=True*, *n_steps=1*, *asymmetric=False*)
Compute the TODI map.

At each voxel an histogram distribution of the local streamlines orientations (TODI) is computed.

Parameters

- **streamlines** (*list of numpy.ndarray*) – List of streamlines.
- **length_weights** (*bool, optional*) – Weights TODI map of each segment's length (default True).

get_mask()

get_sh(*sh_basis*, *sh_order*, *smooth=0.006*, *full_basis=False*)
Spherical Harmonics (SH) coefficients of the TODI map

Compute the SH representation of the TODI map, converting SF to SH with a smoothing factor.

Parameters

- **sh_basis** (*{None, 'tournier07', 'descoteaux07'}*) – `None` for the default DIPY basis, `tournier07` for the Tournier 2007[2] basis, and `descoteaux07` for the Descoteaux 2007[1] basis (`None` defaults to `descoteaux07`).
- **sh_order** (*int*) – Maximum SH order in the SH fit. For *sh_order*, there will be `(sh_order + 1) * (sh_order_2) / 2` SH coefficients (default 4).
- **smooth** (*float, optional*) – Smoothing factor for the conversion, Lambda-regularization in the SH fit (default 0.006).

Returns **todi_sh** – SH representation of the TODI map

Return type ndarray

[2] Tournier J.D., Calamante F. and Connelly A. Robust determination of the fibre orientation distribution in diffusion MRI: Non-negativity constrained super-resolved spherical deconvolution. NeuroImage. 2007;35(4):1459-1472.

[1] Descoteaux, M., Angelino, E., Fitzgibbons, S. and Deriche, R. Regularized, Fast, and Robust Analytical Q-ball Imaging. Magn. Reson. Med. 2007;58:497-510.

References

get_tdi()
Compute the TDI map.

Compute the Tract Density Image (TDI) from the TODI volume.

Returns **tdi** – Tract Density Image

Return type numpy.ndarray (3D)

get_todi()

get_todi_shape()

mask_todi(*mask*)
Mask the TODI map.

Mask the TODI without having to reshape the whole volume all at once (big in memory).

Parameters **mask** (*numpy.ndarray*) – Given volume mask for the TODI map.

normalize_todi_per_voxel(*p_norm=2*)
Normalize TODI map.

Normalize TODI distribution on the sphere for each voxel independently.

Parameters **p_norm** (*int, optional*) – Chosen Norm to normalize.

Returns **todi** – Normalized TODI map.

Return type numpy.ndarray

reshape_to_3d(*img_voxelly_masked*)
Reshape a complex ravelled image to 3D.

Unravel a given unravel mask (1D), image (1D), SH/SF (2D) to its original 3D shape (with a 4D for SH/SF).

Parameters **img_voxelly_masked** (*numpy.ndarray (either in 1D, 2D or 3D)*) – That will be reshaped to 3D (input data shape) accordingly. Necessary for future masking operation.

Returns **unraveled_img** – Unravel volume in x, y, z (, c).

Return type numpy.ndarray (3D, or 4D)

set_todi(*mask*, *todi*)

smooth_todi_dir(*order=2*)
Smooth orientations on the sphere.

Smooth the orientations / distribution on the sphere. Important for priors construction of BST.

Parameters **order** (*int, optional*) – Exponent blurring factor, based on the dot product (default 2).

smooth_todi_spatial(*sigma=0.5*)
Spatial Smoothing of the TODI map.

Blur the TODI map using neighborhood information. Important for priors construction of BST.

Parameters **sigma** (*float, optional*) – Gaussian blurring factor (default 0.5).

1.8.11 scilpy.tractanalysis.todi_util module

scilpy.tractanalysis.todi_util.**compute_vectors_norm**(*vectors*)

scilpy.tractanalysis.todi_util.**generate_mask_indices_1d**(*nb_voxel*, *indices_1d*)

scilpy.tractanalysis.todi_util.**get_dir_to_sphere_id**(*vectors*, *sphere_vertices*)

Find the closest vector on the sphere vertices using a cKDT tree sphere_vertices must be normed (or all with equal norm).

Parameters

- **vectors** (*numpy.ndarray (2D)*) – Vectors representing the direction (x,y,z) of segments.
- **sphere_vertices** (*numpy.ndarray (2D)*) – Vertices of a Dipy sphere object.

Returns **dir_sphere_id** – Sphere indices of the closest sphere direction for each vector

Return type numpy.ndarray (1D)

scilpy.tractanalysis.todi_util.**get_indices_1d**(*volume_shape*, *pts*)

scilpy.tractanalysis.todi_util.**get_segments_dir_and_norm**(*segments*, *seg_mid=None*, *asymmetric=False*)

scilpy.tractanalysis.todi_util.**get_segments_mid_pts_positions**(*segments*)

scilpy.tractanalysis.todi_util.**get_segments_vectors**(*segments*)

scilpy.tractanalysis.todi_util.**get_vectors_dir_and_norm**(*vectors*)

scilpy.tractanalysis.todi_util.**get_vectors_dir_and_norm_rel_to_center**(*vectors*, *seg_mid_pts*)

Evaluates vectors direction and norm by taking into account the orientation and position of segments in relation to the center of voxel

scilpy.tractanalysis.todi_util.**normalize_vectors**(*vectors*)

scilpy.tractanalysis.todi_util.**p_normalize_vectors**(*vectors*, *p*)

scilpy.tractanalysis.todi_util.**psf_from_sphere**(*sphere_vertices*)

scilpy.tractanalysis.todi_util.**streamlines_to_endpoints**(*streamlines*)

Equivalent to streamlines resampling to 2 points (first and last).

Parameters **streamlines** (*list of numpy.ndarray*) – List of streamlines.

Returns **endpoints** – Endpoint array representation with the first and last points.

Return type numpy.ndarray (2D)

scilpy.tractanalysis.todi_util.**streamlines_to_pts_dir_norm**(*streamlines*, *n_steps=1*, *asymmetric=False*)

Evaluate each segment: mid position, direction, length.

Parameters **streamlines** (*list of numpy.ndarray*) – List of streamlines.

Returns

- **seg_mid** (*numpy.ndarray (2D)*) – Mid position (x,y,z) of all streamlines' segments.
- **seg_dir** (*numpy.ndarray (2D)*) – Direction (x,y,z) of all streamlines' segments.

- **seg_norm** (*numpy.ndarray (2D)*) – Length of all streamlines' segments.

scilpy.tractanalysis.todi_util.**streamlines_to_segments**(*streamlines*, *n_steps=1*)
Split streamlines into its segments.

Parameters **streamlines** (*list of numpy.ndarray*) – List of streamlines.

Returns **segments** – Segments array representation with the first and last points.

Return type numpy.ndarray (2D)

1.8.12 scilpy.tractanalysis.tools module

scilpy.tractanalysis.tools.**compute_connectivity**(*indices*, *atlas_data*, *real_labels*, *segmenting_func*)

scilpy.tractanalysis.tools.**compute_streamline_segment**(*orig_strl*, *inter_vox*, *in_vox_idx*, *out_vox_idx*, *points_to_indices*)

scilpy.tractanalysis.tools.**cut_between_masks_streamlines**(*sft*, *binary_mask*, *min_len=0*)
Cut streamlines so their segment are within the bounding box or going from binary mask #1 to binary mask #2. This function erases the data_per_point and data_per_streamline.

Parameters

- **sft** (*StatefulTractogram*) – The sft to cut streamlines (using a single mask with 2 entities) from.
- **binary_mask** (*np.ndarray*) – Boolean array representing the region (must contain 2 entities)
- **min_len** (*float*) – Minimum length from the resulting streamlines.

Returns **new_sft** – New object with the streamlines trimmed within the masks.

Return type StatefulTractogram

scilpy.tractanalysis.tools.**cut_outside_of_mask_streamlines**(*sft*, *binary_mask*, *min_len=0*)
Cut streamlines so their longest segment are within the bounding box or a binary mask. This function erases the data_per_point and data_per_streamline.

Parameters

- **sft** (*StatefulTractogram*) – The sft to cut streamlines (using a single mask with 1 entities) from.
- **binary_mask** (*np.ndarray*) – Boolean array representing the region (must contain 1 entities)
- **min_len** (*float*) – Minimum length from the resulting streamlines.

Returns **new_sft** – New object with the streamlines trimmed within the mask.

Return type StatefulTractogram

scilpy.tractanalysis.tools.**extract_longest_segments_from_profile**(*strl_indices*, *atlas_data*)

scilpy.tractanalysis.tools.**get_point_on_line**(*first_point*, *second_point*, *vox_lower_corner*)

scilpy.tractanalysis.tools.**get_streamline_pt_index**(*points_to_index*, *vox_index*, *from_start=True*)

scilpy.tractanalysis.tools.**intersects_two_rois**(*roi_data_1*, *roi_data_2*, *strl_indices*)
Cut streamlines so their longest segment are within the bounding box or a binary mask. This function keeps the data_per_point and data_per_streamline.

Parameters

- **roi_data_1** (*np.ndarray*) – Boolean array representing the region #1
- **roi_data_2** (*np.ndarray*) – Boolean array representing the region #2
- **strl_indices** (*list of tuple (3xint)*) – 3D indices of the voxel intersected by the streamline

Returns

- **in_strl_idx** (*int*) – index of the first point (of the streamline) to be in the masks
- **out_strl_idx** (*int*) – index of the last point (of the streamline) to be in the masks

scilpy.tractanalysis.tools.**split_heads_tails_kmeans**(*data*)
Split a mask between head and tail with k means.

Parameters **data** (*numpy.ndarray*) – Mask to be split.

Returns

- **mask_1** (*numpy.ndarray*) – "Head" of the mask.
- **mask_2** (*numpy.ndarray*) – "Tail" of the mask.

1.8.13 scilpy.tractanalysis.uncompress module

scilpy.tractanalysis.uncompress.**uncompress**()

1.8.14 scilpy.tractanalysis.uncompress module

scilpy.tractanalysis.uncompress.**uncompress**()

1.9 scilpy.utils package

1.9.1 scilpy.utils.bvec_bval_tools module

class scilpy.utils.bvec_bval_tools.**B0ExtractionStrategy**
Bases: enum.Enum

An enumeration.

ALL = 'all'

FIRST = 'first'

MEAN = 'mean'

scilpy.utils.bvec_bval_tools.**check_b0_threshold**(*force_b0_threshold*, *bvals_min*, *b0_thr=20*)
Check if the minimal bvalue is under zero or over the threshold. If *force_b0_threshold* is true, don't raise an error even if the minimum bvalue is over the threshold.

Parameters

- **force_b0_threshold** (*bool*) – If True, don't raise an error.
- **bvals_min** (*float*) – Minimum bvalue.
- **b0_thr** (*float*) – Maximum bvalue considered as a b0.

Raises ValueError – If the minimal bvalue is over the threshold, and *force_b0_threshold* is False.

scilpy.utils.bvec_bval_tools.**extract_b0**(*dwi*, *b0_mask*, *extract_in_cluster=False*, *strategy=<B0ExtractionStrategy.MEAN: 'mean'>*, *block_size=None*)

Extract a set of b0 volumes from a dwi dataset

Parameters

- **dwi** (*nib.Nifti1Image*) – Original multi-shell volume.
- **b0_mask** (*array of bool*) – Mask over the time dimension (4th) identifying b0 volumes.
- **extract_in_cluster** (*bool*) – Specify to extract b0's in each continuous sets of b0 volumes appearing in the input data.
- **strategy** (*Enum*) – The extraction strategy, of either select the first b0 found, select them all or average them. When used in conjunction with the batch parameter set to True, the strategy is applied individually on each continuous set found.
- **block_size** (*int*) – Load the data using this block size. Useful when the data is too large to be loaded in memory.

Returns **b0_data** – Extracted b0 volumes.

Return type ndarray

scilpy.utils.bvec_bval_tools.**extract_dwi_shell**(*dwi*, *bvals*, *bvecs*, *bvals_to_extract*, *tol=20*, *block_size=None*)

Extracts the DWI volumes that are on specific b-value shells. Many shells can be extracted at once by specifying multiple b-values. The extracted volumes are in the same order as in the original file.

If the b-values of a shell are not all identical, use the –tolerance argument to adjust the accepted interval. For example, a b-value of 2000 and a tolerance of 20 will extract all volumes with a b-values from 1980 to 2020.

Files that are too large to be loaded in memory can still be processed by setting the –block-size argument. A block size of X means that X DWI volumes are loaded at a time for processing.

Parameters

- **dwi** (*nib.Nifti1Image*) – Original multi-shell volume.
- **bvals** (*ndarray*) – The b-values in FSL format.
- **bvecs** (*ndarray*) – The b-vectors in FSL format.
- **bvals_to_extract** (*list of int*) – The list of b-values to extract.
- **tol** (*int*) – The tolerated gap between the b-values to extract and the actual b-values.
- **block_size** (*int*) – Load the data using this block size. Useful when the data is too large to be loaded in memory.

Returns

- **indices** (*ndarray*) – Indices of the volumes corresponding to the provided b-values.
- **shell_data** (*ndarray*) – Volumes corresponding to the provided b-values.
- **output_bvals** (*ndarray*) – Selected b-values.

- **output_bvecs** (*ndarray*) – Selected b-vectors.

scilpy.utils.bvec_bval_tools.**flip_fsl_gradient_sampling**(*bvecs_filename*, *bvecs_flipped_filename*, *axes*)

Flip FSL bvecs on a axis

Parameters

- **bvecs_filename** (*str*) – Bvecs filename
- **bvecs_flipped_filename** (*str*) – Bvecs flipped filename
- **axes** (*list of int*) – List of axes to flip (e.g. [0, 1])

scilpy.utils.bvec_bval_tools.**flip_mrtrix_gradient_sampling**(*gradient_sampling_filename*, *gradient_sampling_flipped_filename*, *axes*)

Flip Mrtrix gradient sampling on a axis

Parameters

- **gradient_sampling_filename** (*str*) – Gradient sampling filename
- **gradient_sampling_flipped_filename** (*str*) – Gradient sampling flipped filename
- **axes** (*list of int*) – List of axes to flip (e.g. [0, 1])

scilpy.utils.bvec_bval_tools.**fsl2mrtrix**(*fsl_bval_filename*, *fsl_bvec_filename*, *mrtrix_filename*)

Convert a fsl dir_grad.bvec/.bval files to mrtrix encoding.b file.

Parameters

- **fsl_bval_filename** (*str*) – path to input fsl bval file.
- **fsl_bvec_filename** (*str*) – path to input fsl bvec file.
- **mrtrix_filename** (*str*) – path to output mrtrix encoding.b file.

scilpy.utils.bvec_bval_tools.**get_shell_indices**(*bvals*, *shell*, *tol=10*)

Get shell indices

Parameters

- **bvals** (*array (N,)*) – array of bvals
- **shell** (*list*) – list of bvals
- **tol** (*int*) – tolerance to accept a bval

Returns

Return type numpy.ndarray where shells are found

scilpy.utils.bvec_bval_tools.**identify_shells**(*bvals*, *threshold=40.0*, *roundCentroids=False*, *sort=False*)

Guessing the shells from the b-values. Returns the list of shells and, for each b-value, the associated shell.

Starting from the first shell as holding the first b-value in bvals, the next b-value is considered on the same shell if it is closer than threshold, or else we consider that it is on another shell. This is an alternative to K-means considering we don't already know the number of shells K.

Note. This function should be added in Dipy soon.

Parameters

- **bvals** (*array (N,)*) – Array of bvals
- **threshold** (*float*) – Limit value to consider that a b-value is on an existing shell. Above this limit, the b-value is placed on a new shell.
- **roundCentroids** (*bool*) – If true will round shell values to the nearest 10.
- **sort** (*bool*) – Sort centroids and shell_indices associated.

Returns

- **centroids** (*array (K)*) – Array of centroids. Each centroid is a b-value representing the shell. K is the number of identified shells.
- **shell_indices** (*array (N,)*) – For each bval, the associated centroid K.

scilpy.utils.bvec_bval_tools.**is_normalized_bvecs**(*bvecs*)

Check if b-vectors are normalized.

Parameters **bvecs** (*(N, 3) array*) – input b-vectors (N, 3) array

Returns

Return type True/False

scilpy.utils.bvec_bval_tools.**mrtrix2fsl**(*mrtrix_filename*, *fsl_bval_filename=None*, *fsl_bvec_filename=None*)

Convert a mrtrix encoding.b file to fsl dir_grad.bvec/.bval files.

Parameters

- **mrtrix_filename** (*str*) – path to mrtrix encoding.b file.
- **fsl_bval_filename** (*str*) – path to the output fsl bval file. Default is mrtrix_filename.bval.
- **fsl_bvec_filename** (*str*) – path to the output fsl bvec file. Default is mrtrix_filename.bvec.

scilpy.utils.bvec_bval_tools.**normalize_bvecs**(*bvecs*, *filename=None*)

Normalize b-vectors

Parameters

- **bvecs** (*(N, 3) array*) – input b-vectors (N, 3) array
- **filename** (*string*) – output filename where to save the normalized bvecs

Returns **bvecs** – normalized b-vectors

Return type (N, 3)

scilpy.utils.bvec_bval_tools.**swap_fsl_gradient_axis**(*bvecs_filename*, *bvecs_swapped_filename*, *axes*)

Swap FSL bvecs

Parameters

- **bvecs_filename** (*str*) – Bvecs filename
- **bvecs_swapped_filename** (*str*) – Bvecs swapped filename
- **axes** (*list of int*) – List of axes to swap (e.g. [0, 1])

scilpy.utils.bvec_bval_tools.**swap_mrtrix_gradient_axis**(*bvecs_filename*, *bvecs_swapped_filename*, *axes*)

Swap MRtrix bvecs

Parameters

- **bvecs_filename** (*str*) – Bvecs filename
- **bvecs_swapped_filename** (*str*) – Bvecs swapped filename
- **axes** (*list of int*) – List of axes to swap (e.g. [0, 1])

1.9.2 scilpy.utils.filenames module

scilpy.utils.filenames.**add_filename_suffix**(*filename*, *suffix*)

This function adds a suffix to the filename, keeping the extension. For example, if filename is test.nii.gz and suffix is "new", the returned name will be test_new.nii.gz

Parameters

- **filename** (*str*) – The full filename, including extension
- **suffix** (*str*) – The suffix to add to the filename

Returns

Return type The completed file name.

scilpy.utils.filenames.**split_name_with_nii**(*filename*)

Returns the clean basename and extension of a file. Means that this correctly manages the ".nii.gz" extensions.

Parameters **filename** (*str*) – The filename to clean

Returns

- **base, ext** (*tuple(str, str)*)
- *Clean basename and the full extension*

1.9.3 scilpy.utils.image module

scilpy.utils.image.**compute_snr**(*dwi*, *bval*, *bvec*, *b0_thr*, *mask*, *noise_mask=None*, *noise_map=None*, *split_shells=False*, *basename=None*, *verbose=False*)

Compute snr

Parameters

- **dwi** (*string*) – Path to the dwi file
- **bvec** (*string*) – Path to the bvec file
- **bval** (*string*) – Path to the bval file
- **b0_thr** (*int*) – Threshold to define b0 minimum value
- **mask** (*string*) – Path to the mask
- **noise_mask** (*string*) – Path to the noise mask
- **noise_map** (*string*) – Path to the noise map
- **basename** (*string*) – Basename used for naming all output files
- **verbose** (*boolean*) – Set to use logging

scilpy.utils.image.**register_image**(*static*, *static_grid2world*, *moving*, *moving_grid2world*, *transformation_type='affine'*, *dwi=None*)

scilpy.utils.image.**transform_anatomy**(*transfo*, *reference*, *moving*, *filename_to_save*, *interp='linear'*, *keep_dtype=False*)

Apply transformation to an image using Dipy's tool

Parameters

- **transfo** (*numpy.ndarray*) – Transformation matrix to be applied
- **reference** (*str*) – Filename of the reference image (target)
- **moving** (*str*) – Filename of the moving image
- **filename_to_save** (*str*) – Filename of the output image
- **interp** (*string, either 'linear' or 'nearest'*) – the type of interpolation to be used, either 'linear' (for k-linear interpolation) or 'nearest' for nearest neighbor
- **keep_dtype** (*bool*) – If True, keeps the data_type of the input moving image when saving the output image

scilpy.utils.image.**transform_dwi**(*reg_obj*, *static*, *dwi*, *interpolation='linear'*)

Iteratively apply transformation to 4D image using Dipy's tool

Parameters

- **reg_obj** (*AffineMap*) – Registration object from Dipy returned by AffineMap
- **static** (*numpy.ndarray*) – Target image data
- **dwi** (*numpy.ndarray*) – 4D numpy array containing a scalar in each voxel (moving image data)
- **interpolation** (*string, either 'linear' or 'nearest'*) – the type of interpolation to be used, either 'linear' (for k-linear interpolation) or 'nearest' for nearest neighbor

1.9.4 scilpy.utils.metrics_tools module

scilpy.utils.metrics_tools.**compute_lesion_stats**(*map_data*, *lesion_atlas*, *single_label=True*, *voxel_sizes=[1.0, 1.0, 1.0]*, *min_lesion_vol=7*, *precomputed_lesion_labels=None*)

Returns information related to lesion inside of a binary mask or voxel labels map (bundle, for tractometry).

Parameters

- **map_data** (*np.ndarray*) – Either a binary mask (uint8) or a voxel labels map (int16).
- **lesion_atlas** (*np.ndarray (3)*) – Labelled atlas of lesion. Should be int16.
- **single_label** (*boolean*) – If true, does not add an extra layer for number of labels.
- **voxel_sizes** (*np.ndarray (3)*) – If not specified, returns voxel count (instead of volume)
- **min_lesion_vol** (*float*) – Minimum lesion volume in mm3 (default: 7, cross-shape).
- **precomputed_lesion_labels** (*np.ndarray (N)*) – For connectivity analysis, when the unique lesion labels are known, provided a pre-computed list of labels save computation.

Returns **lesion_load_dict** – For each label, volume and lesion count

Return type dict

scilpy.utils.metrics_tools.**get_bundle_metrics_mean_std**(*streamlines*, *metrics_files*, *distance_values*, *correlation_values*, *density_weighting=True*)

Returns the mean value of each metric for the whole bundle, only considering voxels that are crossed by streamlines. The mean values are weighted by the number of streamlines crossing a voxel by default. If false, every voxel traversed by a streamline has the same weight.

Parameters

- **streamlines** (*list of numpy.ndarray*) – Input streamlines under which to compute stats.
- **metrics_files** (*sequence*) – list of nibabel objects representing the metrics files
- **density_weighting** (*bool*) – weigh by the mean by the density of streamlines going through the voxel

Returns **stats** – list of tuples where the first element of the tuple is the mean of a metric, and the second element is the standard deviation, for each metric.

Return type list

scilpy.utils.metrics_tools.**get_bundle_metrics_mean_std_per_point**(*streamlines*, *bundle_name*, *metrics*, *labels*, *distance_values=None*, *correlation_values=None*, *density_weighting=False*)

Compute the mean and std PER POiNT of the bundle for every given metric.

Parameters

- **streamlines** (*list of numpy.ndarray*) – Input streamlines under which to compute stats.
- **bundle_name** (*str*) – Name of the bundle. Will be used as a key in the dictionary.
- **metrics** (*sequence*) – list of nibabel objects representing the metrics files
- **labels** (*np.ndarray*) – List of labels obtained with scil_label_and_distance_maps.py
- **distance_values** (*np.ndarray*) – List of distances obtained with scil_compute_bundle_voxel_label_map.py
- **correlation_values** (*np.ndarray*) – List of correlations obtained with scil_compute_bundle_voxel_label_map.py
- **density_weighting** (*bool*) – If true, weight statistics by the number of streamlines passing through each voxel. [False]
- **distance_weighting** (*bool*) – If true, weight statistics by the inverse of the distance between a streamline and the centroid.

Returns

Return type stats

scilpy.utils.metrics_tools.**get_bundle_metrics_profiles**(*sft*, *metrics_files*)

Returns the profile of each metric along each streamline from a sft. This is used to create tract profiles.

Parameters

- **sft** (*StatefulTractogram*) – Input bundle under which to compute profile.
- **metrics_files** (*sequence*) – list of nibabel objects representing the metrics files

Returns **profiles_values** – list of profiles for each streamline, per metric given

Return type list

scilpy.utils.metrics_tools.**get_roi_metrics_mean_std**(*density_map*, *metrics_files*)

Returns the mean and standard deviation of each metric, using the provided density map. This can be a binary mask, or contain weighted values between 0 and 1.

Parameters

- **density_map** (*ndarray*) – 3D numpy array containing a density map.
- **metrics_files** (*sequence*) – list of nibabel objects representing the metrics files.

Returns **stats** – list of tuples where the first element of the tuple is the mean of a metric, and the second element is the standard deviation.

Return type list

scilpy.utils.metrics_tools.**plot_metrics_stats**(*means*, *stds*, *title=None*, *xlabel=None*, *ylabel=None*, *figlabel=None*, *fill_color=None*, *display_means=False*)

Plots the mean of a metric along n points with the standard deviation.

Parameters

- **mean** (*Numpy 1D (or 2D) array of size n*) – Mean of the metric along n points.
- **std** (*Numpy 1D (or 2D) array of size n*) – Standard deviation of the metric along n points.
- **title** (*string*) – Title of the figure.
- **xlabel** (*string*) – Label of the X axis.
- **ylabel** (*string*) – Label of the Y axis (suggestion: the metric name).
- **figlabel** (*string*) – Label of the figure (only metadata in the figure object returned).
- **fill_color** (*string*) – Hexadecimal RGB color filling the region between mean $\pm$ std. The hexadecimal RGB color should be formatted as #RRGGBB
- **display_means** (*bool*) – Display the subjects means as semi-transparent line

Returns

Return type The figure object.

scilpy.utils.metrics_tools.**weighted_mean_std**(*weights*, *data*)

Returns the weighted mean and standard deviation of the data.

Parameters

- **weights** (*ndarray*) – a ndarray containing the weighting factor
- **data** (*ndarray*) – the ndarray containing the data for which the stats are desired

Returns **stats** – a tuple containing the mean and standard deviation of the data

Return type tuple

1.9.5 scilpy.utils.streamlines module

scilpy.utils.streamlines.**compress_sft**(*sft*, *tol_error=0.01*)
Compress a stateful tractogram. Uses Dipy's compress_streamlines, but deals with space better.

Dipy's description: The compression consists in merging consecutive segments that are nearly collinear. The merging is achieved by removing the point the two segments have in common.

The linearization process [Presseau15] ensures that every point being removed are within a certain margin (in mm) of the resulting streamline. Recommendations for setting this margin can be found in [Presseau15] (in which they called it tolerance error).

The compression also ensures that two consecutive points won't be too far from each other (precisely less or equal than ‘max_segment_length‘mm). This is a tradeoff to speed up the linearization process [Rheault15]. A low value will result in a faster linearization but low compression, whereas a high value will result in a slower linearization but high compression.

[Presseau C. et al., A new compression format for fiber tracking datasets, NeuroImage, no 109, 73-83, 2015.]

Parameters

- **sft** (*StatefulTractogram*) – The sft to compress.
- **tol_error** (*float (optional)*) – Tolerance error in mm (default: 0.01). A rule of thumb is to set it to 0.01mm for deterministic streamlines and 0.1mm for probabilitic streamlines.

Returns **compressed_sft**

Return type StatefulTractogram

scilpy.utils.streamlines.**cut_invalid_streamlines**(*sft*)
Cut streamlines so their longest segment are within the bounding box. This function keeps the data_per_point and data_per_streamline.

Parameters **sft** (*StatefulTractogram*) – The sft to remove invalid points from.

Returns

- **new_sft** (*StatefulTractogram*) – New object with the invalid points removed from each streamline.
- **cutting_counter** (*int*) – Number of streamlines that were cut.

scilpy.utils.streamlines.**filter_tractogram_data**(*tractogram*, *streamline_ids*)
Filter tractogram according to streamline ids and keep the data

tractogram: StatefulTractogram Tractogram containing the data to be filtered

streamline_ids: array_like List of streamline ids the data corresponds to

new_tractogram: Tractogram or StatefulTractogram Returns a new tractogram with only the selected streamlines and data

scilpy.utils.streamlines.**get_color_streamlines_along_length**(*sft*, *colormap='jet'*)
Color streamlines according to their length.

Parameters

- **sft** (*StatefulTractogram*) – The tractogram that contains the list of streamlines to be colored
- **colormap** (*str*) – The colormap to use.

Returns **color** – An array of shape (nb_streamlines, 3) containing the RGB values of streamlines

Return type np.ndarray

scilpy.utils.streamlines.**uniformize_bundle_sft**(*sft*, *axis=None*, *ref_bundle=None*, *swap=False*)

Uniformize the streamlines in the given tractogram.

Parameters

- **sft** (*StatefulTractogram*) – The tractogram that contains the list of streamlines to be uniformized
- **axis** (*int, optional*) – Orient endpoints in the given axis
- **ref_bundle** (*streamlines*) – Orient endpoints the same way as this bundle (or centroid)
- **swap** (*boolean, optional*) – Swap the orientation of streamlines

scilpy.utils.streamlines.**uniformize_bundle_sft_using_mask**(*sft*, *mask*, *swap=False*)

Uniformize the streamlines in the given tractogram so head is closer to to a region of interest.

Parameters

- **sft** (*StatefulTractogram*) – The tractogram that contains the list of streamlines to be uniformized
- **mask** (*np.ndarray*) – Mask to use as a reference for the ROI.
- **swap** (*boolean, optional*) – Swap the orientation of streamlines

1.9.6 scilpy.utils.util module

scilpy.utils.util.**compute_distance_barycenters**(*ref_1*, *ref_2*, *ref_2_transfo*)

Compare the barycenter (center of volume) of two reference object. The provided transformation will move the reference #2 and return the distance before and after transformation.

Parameters

- **ref_1** (*reference object*) – Any type supported by the sft as reference (e.g .nii of .trk).
- **ref_2** (*reference object*) – Any type supported by the sft as reference (e.g .nii of .trk).
- **ref_2_transfo** (*np.ndarray*) – Transformation that modifies the barycenter of ref_2.

Returns **distance** – return a tuple containing the distance before and after the transformation.

Return type float or tuple (2,)

scilpy.utils.util.**is_float**(*value*)

Returns True if the argument can be casted to a float.

scilpy.utils.util.**str_to_index**(*axis*)

Convert x y z axis string to 0 1 2 axis index

Parameters **axis** (*str*) – Axis value (x, y or z)

Returns **index** – Axis index

Return type int or None

scilpy.utils.util.**voxel_to_world**(*coord*, *affine*)
Takes a n dimensionnal voxel coordinate and returns its 3 first coordinates transformed to world space from a given voxel to world affine transformation.

scilpy.utils.util.**world_to_voxel**(*coord*, *affine*)
Takes a n dimensionnal world coordinate and returns its 3 first coordinates transformed to voxel space from a given voxel to world affine transformation.

1.10 scilpy.viz package

1.10.1 scilpy.viz.gradient_sampling module

scilpy.viz.gradient_sampling.**build_ms_from_shell_idx**(*bvecs*, *shell_idx*)
Get bvecs from indexes

Parameters

- **bvecs** (*numpy.ndarray*) – bvecs
- **shell_idx** (*numpy.ndarray*) – index for each bval

Returns **ms** – bvecs for each bval

Return type list of numpy.ndarray

scilpy.viz.gradient_sampling.**plot_each_shell**(*ms*, *centroids*, *plot_sym_vecs=True*, *use_sphere=True*, *same_color=False*, *rad=0.025*, *opacity=1.0*, *ofile=None*, *ores=(300, 300)*)
Plot each shell

Parameters

- **ms** (*list of lists of numpy.ndarray*) – bvecs for each bval: one list per shell.
- **centroids** (*list of ints*) – List of shells to plot.
- **plot_sym_vecs** (*boolean*) – Plot symmetrical vectors
- **use_sphere** (*boolean*) – rendering of the sphere
- **same_color** (*boolean*) – use same color for all shell
- **rad** (*float*) – radius of each point
- **opacity** (*float*) – opacity for the shells
- **ofile** (*str*) – output filename
- **ores** (*tuple*) – resolution of the output png

scilpy.viz.gradient_sampling.**plot_proj_shell**(*ms*, *use_sym=True*, *use_sphere=True*, *same_color=False*, *rad=0.025*, *opacity=1.0*, *ofile=None*, *ores=(300, 300)*)
Plot each shell

Parameters

- **ms** (*list of lists of numpy.ndarray*) – bvecs for each bval: one list per shell.
- **use_sym** (*boolean*) – Plot symmetrical vectors
- **use_sphere** (*boolean*) – rendering of the sphere

- **same_color** (*boolean*) – use same color for all shell
- **rad** (*float*) – radius of each point
- **opacity** (*float*) – opacity for the shells
- **ofile** (*str*) – output filename
- **ores** (*tuple*) – resolution of the output png

1.10.2 scilpy.viz.screenshot module

scilpy.viz.screenshot.**display_slices**(*volume_actor*, *slices*, *output_filename*, *axis_name*, *view_position*, *focal_point*, *peaks_actor=None*, *streamlines_actor=None*)

CHAPTER 2

Scripts

2.1 __init__.py

2.2 scil_add_tracking_mask_to_pft_maps.py

```
usage: __main__.py [-h] [-f]
                   map_include map_exclude additional_mask map_include_corr
                   map_exclude_corr

Modify PFT maps to allow PFT tracking in given mask (e.g edema).

positional arguments:
  map_include       PFT map include.
  map_exclude       PFT map exclude.
  additional_mask   Allow PFT tracking in this mask.
  map_include_corr  Corrected PFT map include output file name.
  map_exclude_corr  Corrected PFT map exclude output file name.

optional arguments:
  -h, --help        show this help message and exit
  -f                Force overwriting of the output files.
```

2.3 scil_analyse_lesions_load.py

```
usage: __main__.py [-h]
                   [--bundle BUNDLE | --bundle_mask BUNDLE_MASK | --bundle_labels_map␣
↪BUNDLE_LABELS_MAP]
                   [--min_lesion_vol MIN_LESION_VOL] [--out_lesion_atlas FILE]
                   [--out_lesion_stats FILE] [--out_streamlines_stats FILE]
                   [--indent INDENT] [--sort_keys] [-f]
```

(continues on next page)

(continued from previous page)

```
                    [--reference REFERENCE]
                    in_lesion out_json

This script will output informations about lesion load in bundle(s).
The input can either be streamlines, binary bundle map, or a bundle voxel
label map.

To be considered a valid lesion, the lesion volume must be at least
min_lesion_vol mm3. This avoid the detection of thousand of single voxel
lesions if an automatic lesion segmentation tool is used.

positional arguments:
  in_lesion             Binary mask of the lesion(s) (.nii.gz).
  out_json              Output file for lesion information (.json).

optional arguments:
  -h, --help            show this help message and exit
  --bundle BUNDLE       Path of the bundle file (.trk).
  --bundle_mask BUNDLE_MASK
                        Path of the bundle binary mask (.nii.gz).
  --bundle_labels_map BUNDLE_LABELS_MAP
                        Path of the bundle labels map (.nii.gz).
  --min_lesion_vol MIN_LESION_VOL
                        Minimum lesion volume in mm3 [7].
  --out_lesion_atlas FILE
                        Save the labelized lesion(s) map (.nii.gz).
  --out_lesion_stats FILE
                        Save the lesion-wise volume measure (.json).
  --out_streamlines_stats FILE
                        Save the lesion-wise streamline count (.json).
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.4 scil_apply_bias_field_on_dwi.py

```
usage: __main__.py [-h] [--mask MASK] [-f] in_dwi in_bias_field out_name

Apply bias field correction to DWI. This script doesn't compute the bias
field itself. It ONLY applies an existing bias field. Use the ANTs
N4BiasFieldCorrection executable to compute the bias field

positional arguments:
  in_dwi         DWI Nifti image.
  in_bias_field  Bias field Nifti image.
  out_name       Corrected DWI Nifti image.

optional arguments:
  -h, --help     show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --mask MASK    Apply bias field correction only in the region defined by the mask.
  -f             Force overwriting of the output files.
```

2.5 scil_apply_transform_to_bvecs.py

```
usage: __main__.py [-h] [--inverse] [-f] in_bvecs in_transfo out_bvecs

Transform bvecs using an affine/rigid transformation.

positional arguments:
  in_bvecs    Path of the bvec file, in FSL format
  in_transfo  Path of the file containing the 4x4
              transformation, matrix (.txt, .npy or .mat).
  out_bvecs   Output filename of the transformed bvecs.

optional arguments:
  -h, --help  show this help message and exit
  --inverse   Apply the inverse transformation.
  -f          Force overwriting of the output files.
```

2.6 scil_apply_transform_to_hdf5.py

```
usage: __main__.py [-h] [--inverse] [--in_deformation IN_DEFORMATION]
                   [--reverse_operation] [--cut_invalid]
                   [--reference REFERENCE] [-f]
                   in_hdf5 in_target_file in_transfo out_hdf5

Transform connectivity hdf5 (.h5) using an affine/rigid transformation and
nonlinear deformation (optional).

For more information on how to use the registration script, follow this link:
https://scilpy.readthedocs.io/en/latest/documentation/tractogram_registration.html

Example:
To apply transform from ANTS to tractogram. If the ANTS commands was
MOVING->REFERENCE, this will bring a tractogram from MOVING->REFERENCE
scil_apply_transform_to_tractogram.py ${MOVING_FILE} ${REFERENCE_FILE}
                                      0GenericAffine.mat ${OUTPUT_NAME}
                                      --inverse
                                      --in_deformation 1InverseWarp.nii.gz

If the ANTS commands was MOVING->REFERENCE, this will bring a tractogram
from REFERENCE->MOVING
scil_apply_transform_to_tractogram.py ${MOVING_FILE} ${REFERENCE_FILE}
                                      0GenericAffine.mat ${OUTPUT_NAME}
                                      --in_deformation 1Warp.nii.gz
                                      --reverse_operation

positional arguments:
  in_hdf5               Path of the tractogram to be transformed.
  in_target_file        Path of the reference target file (.trk or .nii).
```

(continues on next page)

(continued from previous page)

```
  in_transfo            Path of the file containing the 4x4
                        transformation, matrix (.txt, .npy or .mat).
  out_hdf5              Output tractogram filename (transformed data).

optional arguments:
  -h, --help            show this help message and exit
  --inverse             Apply the inverse linear transformation.
  --in_deformation IN_DEFORMATION
                        Path to the file containing a deformation field.
  --reverse_operation   Apply the transformation in reverse (see doc), warp first,␣
↪then linear.
  --cut_invalid         Cut invalid streamlines rather than removing them.
                        Keep the longest segment only.
                        By default, invalid streamline are removed.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.7 scil_apply_transform_to_image.py

```
usage: __main__.py [-h] [--inverse] [--keep_dtype] [-f]
                   in_file in_target_file in_transfo out_name

Transform Nifti (.nii.gz) using an affine/rigid transformation.

For more information on how to use the registration script, follow this link:
https://scilpy.readthedocs.io/en/latest/documentation/tractogram_registration.html

positional arguments:
  in_file         Path of the file to be transformed (nii or nii.gz)
  in_target_file  Path of the reference target file (.nii.gz).
  in_transfo      Path of the file containing the 4x4
                  transformation, matrix (.txt, .npy or .mat).
  out_name        Output filename of the transformed data.

optional arguments:
  -h, --help      show this help message and exit
  --inverse       Apply the inverse transformation.
  --keep_dtype    If True, keeps the data_type of the input image (in_file) when␣
↪saving the output image (out_name).
  -f              Force overwriting of the output files.
```

2.8 scil_apply_transform_to_surface.py

```
usage: __main__.py [-h] [--ants_warp ANTS_WARP] [-f]
                   in_surface ants_affine out_surface

Script to load and transform a surface (FreeSurfer or VTK supported),
This script is using ANTs transform (affine.txt, warp.nii.gz).
```

(continues on next page)

(continued from previous page)

```
Best usage with ANTs from T1 to b0:
> ConvertTransformFile 3 output0GenericAffine.mat vtk_transfo.txt --hm
> scil_transform_surface.py lh_white_lps.vtk affine.txt lh_white_b0.vtk\
    --ants_warp warp.nii.gz

The input surface needs to be in *T1 world LPS* coordinates
(aligned over the T1 in MI-Brain).
The resulting surface should be aligned *b0 world LPS* coordinates
(aligned over the b0 in MI-Brain).

positional arguments:
  in_surface            Input surface (.vtk).
  ants_affine           Affine transform from ANTs (.txt or .mat).
  out_surface           Output surface (.vtk).

optional arguments:
  -h, --help            show this help message and exit
  --ants_warp ANTS_WARP
                        Warp image from ANTs (NIfTI format).
  -f                    Force overwriting of the output files.

References:
[1] St-Onge, E., Daducci, A., Girard, G. and Descoteaux, M. 2018.
    Surface-enhanced tractography (SET). NeuroImage.
```

2.9 scil_apply_transform_to_tractogram.py

```
usage: __main__.py [-h] [--inverse] [--in_deformation IN_DEFORMATION]
                   [--reverse_operation]
                   [--cut_invalid | --remove_invalid | --keep_invalid]
                   [--no_empty] [--reference REFERENCE] [-f] [-v]
                   in_moving_tractogram in_target_file in_transfo
                   out_tractogram

Transform tractogram using an affine/rigid transformation and nonlinear
deformation (optional).

For more information on how to use the registration script, follow this link:
https://scilpy.readthedocs.io/en/latest/documentation/tractogram_registration.html

Applying transformation to tractogram can lead to invalid streamlines (out of
the bounding box), three strategies are available:
1) default, crash at saving if invalid streamlines are present
2) --keep_invalid, save invalid streamlines. Leave it to the user to run
    scil_remove_invalid_streamlines.py if needed.
3) --remove_invalid, automatically remove invalid streamlines before saving.
    Should not remove more than a few streamlines.
4) --cut_invalid, automatically cut invalid streamlines before saving.

Example:
To apply transform from ANTS to tractogram. If the ANTS commands was
MOVING->REFERENCE, this will bring a tractogram from MOVING->REFERENCE
scil_apply_transform_to_tractogram.py ${MOVING_FILE} ${REFERENCE_FILE}
                                      0GenericAffine.mat ${OUTPUT_NAME}
```

(continues on next page)

(continued from previous page)

```
                                        --inverse
                                        --in_deformation 1InverseWarp.nii.gz

If the ANTS commands was MOVING->REFERENCE, this will bring a tractogram
from REFERENCE->MOVING
scil_apply_transform_to_tractogram.py ${MOVING_FILE} ${REFERENCE_FILE}
                                        0GenericAffine.mat ${OUTPUT_NAME}
                                        --in_deformation 1Warp.nii.gz
                                        --reverse_operation

positional arguments:
  in_moving_tractogram  Path of the tractogram to be transformed.
                        Bounding box validity will not be checked (could contain␣
↪invalid streamlines).
  in_target_file        Path of the reference target file (trk or nii).
  in_transfo            Path of the file containing the 4x4
                        transformation, matrix (.txt, .npy or .mat).
  out_tractogram        Output tractogram filename (transformed data).

optional arguments:
  -h, --help            show this help message and exit
  --inverse             Apply the inverse linear transformation.
  --in_deformation IN_DEFORMATION
                        Path to the file containing a deformation field.
  --reverse_operation   Apply the transformation in reverse (see doc),warp first,␣
↪then linear.
  --cut_invalid         Cut invalid streamlines rather than removing them.
                        Keep the longest segment only.
  --remove_invalid      Remove the streamlines landing out of the bounding box.
  --keep_invalid        Keep the streamlines landing out of the bounding box.
  --no_empty            Do not write file if there is no streamline.
                        You may save an empty file if you use remove_invalid.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.10 scil_assign_custom_color_to_tractogram.py

```
usage: __main__.py [-h] [--out_colorbar OUT_COLORBAR] [--horizontal_cbar]
                   [--use_dps DPS_KEY | --use_dpp DPP_KEY | --load_dps DPS_KEY | --
↪load_dpp DPP_KEY | --from_anatomy FILE | --along_profile]
                   [--colormap COLORMAP] [--min_range MIN_RANGE]
                   [--max_range MAX_RANGE] [--log] [--LUT FILE]
                   [--reference REFERENCE] [-f]
                   in_tractogram out_tractogram

The script uses scalars from an anatomy, data_per_point or data_per_streamline
(e.g commit_weights) to visualize them on the streamlines.
Saves the RGB values in the data_per_point (color_x, color_y, color_z).

If called with .tck, the output will always be .trk, because data_per_point has
no equivalent in tck file.
```

(continues on next page)

(continued from previous page)

```
The usage of --use_dps, --use_dpp and --from_anatomy is more complex. It maps
the raw values from these sources to RGB using a colormap.
    --use_dps: total nbr of streamlines of the tractogram = len(streamlines)
    --use_dpp: total nbr of points of the tractogram = len(streamlines._data)

A minimum and a maximum range can be provided to clip values. If the range of
values is too large for intuitive visualization, a log transform can be
applied.

If the data provided from --use_dps, --use_dpp and --from_anatomy are integer
labels, they can be mapped using a LookUp Table (--LUT).
The file provided as a LUT should be either .txt or .npy and if the
size is N=20, then the data provided should be between 1-20.

Example: Use --from_anatomy with a voxel labels map (values from 1-20) with a
text file containing 20 p-values to map p-values to the bundle for
visualisation.

A custom colormap can be provided using --colormap. It should be a string
containing a colormap name OR multiple Matplotlib named colors separated by -.
The colormap used for mapping values to colors can be saved to a png/jpg image
using the --out_colorbar option.

The script can also be used to color streamlines according to their length
using the --along_profile option. The streamlines must be uniformized.

positional arguments:
  in_tractogram         Input tractogram (.trk or .tck).
  out_tractogram        Output tractogram (.trk or .tck).

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Colorbar Options:
  --out_colorbar OUT_COLORBAR
                        Optional output colorbar (.png, .jpg or any format supported␣
↪by matplotlib).
  --horizontal_cbar     Draw horizontal colorbar (vertical by default).

Coloring Methods:
  --use_dps DPS_KEY     Use the data_per_streamline (scalar) for coloring,
                        linear scaling from min-max, e.g. commit_weights.
  --use_dpp DPP_KEY     Use the data_per_point (scalar) for coloring,
                        linear scaling from min-max.
  --load_dps DPS_KEY    Load data per streamline (scalar) for coloring
  --load_dpp DPP_KEY    Load data per point (scalar) for coloring
  --from_anatomy FILE   Use the voxel data for coloring,
                        linear scaling from minmax.
  --along_profile       Color streamlines according to each point positionalong its␣
↪length.
                        Must be uniformized head/tail.
```

(continues on next page)

(continued from previous page)

```
Coloring Options:
  --colormap COLORMAP   Select the colormap for colored trk (dps/dpp) [jet].
                        Use two Matplotlib named color separeted by a - to create␣
↪your own colormap.
  --min_range MIN_RANGE
                        Set the minimum value when using dps/dpp/anatomy.
  --max_range MAX_RANGE
                        Set the maximum value when using dps/dpp/anatomy.
  --log                 Apply a base 10 logarithm for colored trk (dps/dpp).
  --LUT FILE            If the dps/dpp or anatomy contain integer labels, the value␣
↪will be substituted.
                        If the LUT has 20 elements, integers from 1-20 in the data␣
↪will be
                        replaced by the value in the file (.npy or .txt)
```

2.11 scil_assign_uniform_color_to_tractograms.py

```
usage: __main__.py [-h] [--fill_color FILL_COLOR | --dict_colors DICT_COLORS]
                   [--out_suffix OUT_SUFFIX | --out_tractogram OUT_TRACTOGRAM]
                   [--reference REFERENCE] [-f]
                   in_tractograms [in_tractograms ...]

Assign an hexadecimal RGB color to a Trackvis TRK tractogram.
The hexadecimal RGB color should be formatted as 0xRRGGBB or
"#RRGGBB".

Saves the RGB values in the data_per_point (color_x, color_y, color_z).

If called with .tck, the output will always be .trk, because data_per_point has
no equivalent in tck file.

positional arguments:
  in_tractograms        Input tractograms (.trk or .tck).

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Coloring Methods:
  --fill_color FILL_COLOR
                        Can be either hexadecimal (ie. "#RRGGBB" or 0xRRGGBB).
  --dict_colors DICT_COLORS
                        Dictionnary mapping basename to color.
                        Same convention as --fill_color.

Output options:
  --out_suffix OUT_SUFFIX
                        Specify suffix to append to input basename.
  --out_tractogram OUT_TRACTOGRAM
                        Output filename of colored tractogram (.trk).
                        Cannot be used with --dict_colors.
```

2.12 scil_clean_qbx_clusters.py

```
usage: __main__.py [-h] [--out_accepted_dir OUT_ACCEPTED_DIR]
                   [--out_rejected_dir OUT_REJECTED_DIR]
                   [--min_cluster_size MIN_CLUSTER_SIZE]
                   [--background_opacity BACKGROUND_OPACITY]
                   [--background_linewidth BACKGROUND_LINEWIDTH]
                   [--clusters_linewidth CLUSTERS_LINEWIDTH]
                   [--reference REFERENCE] [-f] [-v] [--no_bbox_check]
                   in_bundles [in_bundles ...] out_accepted out_rejected

    Render clusters sequentially to either accept or reject them based on
    visual inspection. Useful for cleaning bundles for RBx, BST or for figures.
    The VTK window does not handle well opacity of streamlines, this is a
    normal rendering behavior.
    Often use in pair with scil_compute_qbx.py.

    Key mapping:
    - a/A: accept displayed clusters
    - r/R: reject displayed clusters
    - z/Z: Rewing one element
    - c/C: Stop rendering of the background concatenation of streamlines
    - q/Q: Early window exist, everything remaining will be rejected

positional arguments:
  in_bundles            List of the clusters filename.
  out_accepted          Filename of the concatenated accepted clusters.
  out_rejected          Filename of the concatenated rejected clusters.

optional arguments:
  -h, --help            show this help message and exit
  --out_accepted_dir OUT_ACCEPTED_DIR
                        Directory to save all accepted clusters separately.
  --out_rejected_dir OUT_REJECTED_DIR
                        Directory to save all rejected clusters separately.
  --min_cluster_size MIN_CLUSTER_SIZE
                        Minimum cluster size for consideration [1].Must be at least 1.
  --background_opacity BACKGROUND_OPACITY
                        Opacity of the background streamlines.Keep low between 0 and␣
↪0.5 [0.1].
  --background_linewidth BACKGROUND_LINEWIDTH
                        Linewidth of the background streamlines [1].
  --clusters_linewidth CLUSTERS_LINEWIDTH
                        Linewidth of the current cluster [1].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).
```

2.13 scil_combine_labels.py

```
usage: __main__.py [-h] --volume_ids VOLUME_IDS [VOLUME_IDS ...]
                   [--out_labels_ids OUT_LABELS_IDS [OUT_LABELS_IDS ...] |
                   --unique | --group_in_m] [--background BACKGROUND]
                   [--merge_groups] [-f]
                   output

    Script to combine labels from multiple volumes.
    If there is overlap, it will overwrite them based on the input order.

    >>> scil_combine_labels.py out_labels.nii.gz
            --volume_ids animal_labels.nii 20
            --volume_ids DKT_labels.nii.gz 44 53
            --out_labels_indices 20 44 53
    >>> scil_combine_labels.py slf_labels.nii.gz
            --volume_ids a2009s_aseg.nii.gz all
            --volume_ids clean/s1__DKT.nii.gz 1028 2028

positional arguments:
  output                Combined labels volume output.

optional arguments:
  -h, --help            show this help message and exit
  --volume_ids VOLUME_IDS [VOLUME_IDS ...]
                        List of volumes directly followed by their labels:
                          --volume_ids atlasA  id1a id2a
                          --volume_ids atlasB  id1b id2b ...
                          "all" can be used instead of id numbers.
  --out_labels_ids OUT_LABELS_IDS [OUT_LABELS_IDS ...]
                        List of labels indices for output images.
  --unique              If set, output id with unique labels, excluding first␣
↪background value.
  --group_in_m          Add (x * 10 000) to each volume labels, where x is the input␣
↪volume order number.
  --background BACKGROUND
                        Background id, excluded from output [0],
                         the value is used as output background value.
  --merge_groups        Each group from the --volume_ids option will be merged as a␣
↪single labels.
  -f                    Force overwriting of the output files.

    References:
        [1] Al-Sharif N.B., St-Onge E., Vogel J.W., Theaud G.,
            Evans A.C. and Descoteaux M. OHBM 2019.
            Surface integration for connectome analysis in age prediction.
```

2.14 scil_compare_connectivity.py

```
usage: __main__.py [-h] --in_g1 IN_G1 [IN_G1 ...] --in_g2 IN_G2 [IN_G2 ...]
                   [--tail {left,right,both}] [--paired]
                   [--fdr | --bonferroni] [--p_threshold THRESH OUT_FILE]
                   [--filtering_mask FILTERING_MASK] [-v] [-f]
                   out_pval_matrix
```

(continues on next page)

(continued from previous page)

```
Performs a network-based statistical comparison for populations g1 and g2. The
output is a matrix of the same size as the input connectivity matrices, with
p-values at each edge.
All input matrices must have the same shape (NxN). For paired t-test, both
groups must have the same number of observations.

For example, if you have streamline count weighted matrices for a MCI and a
control group and you want to investiguate differences in their connectomes:
>>> scil_compare_connectivity.py pval.npy --g1 MCI/*_sc.npy --g2 CTL/*_sc.npy

--filtering_mask will simply multiply the binary mask to all input
matrices before performing the statistical comparison. Reduces the number of
statistical tests, useful when using --fdr or --bonferroni.

positional arguments:
  out_pval_matrix       Output matrix (.npy) containing the edges p-value.

optional arguments:
  -h, --help            show this help message and exit
  --in_g1 IN_G1 [IN_G1 ...]
                        List of matrices for the first population (.npy).
  --in_g2 IN_G2 [IN_G2 ...]
                        List of matrices for the second population (.npy).
  --tail {left,right,both}
                        Enables specification of an alternative hypothesis:
                        left: mean of g1 < mean of g2,
                        right: mean of g2 < mean of g1,
                        both: both means are not equal (default).
  --paired              Use paired sample t-test instead of population t-test.
                        --in_g1 and --in_g2 must be ordered the same way.
  --fdr                 Perform a false discovery rate (FDR) correction for the p-
↪values.
                        Uses the number of non-zero edges as number of tests (value␣
↪between 0.01 and 0.1).
  --bonferroni          Perform a Bonferroni correction for the p-values.
                        Uses the number of non-zero edges as number of tests.
  --p_threshold THRESH OUT_FILE
                        Threshold the final p-value matrix and save the binary matrix␣
↪(.npy).
  --filtering_mask FILTERING_MASK
                        Binary filtering mask (.npy) to apply before computing the␣
↪measures.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

[1] Rubinov, Mikail, and Olaf Sporns. "Complex network measures of brain
    connectivity: uses and interpretations." Neuroimage 52.3 (2010):
    1059-1069.
[2] Zalesky, Andrew, Alex Fornito, and Edward T. Bullmore. "Network-based
    statistic: identifying differences in brain networks." Neuroimage 53.4
    (2010): 1197-1207.
```

2.15 scil_compress_streamlines.py

```
usage: __main__.py [-h] [-e ERROR_RATE] [-f] in_tractogram out_tractogram

Compress tractogram by removing collinear (or almost) points.

The compression threshold represents the maximum distance (in mm) to the
original position of the point.

positional arguments:
  in_tractogram   Path of the input tractogram file (trk or tck).
  out_tractogram  Path of the output tractogram file (trk or tck).

optional arguments:
  -h, --help      show this help message and exit
  -e ERROR_RATE   Maximum compression distance in mm [0.1].
  -f              Force overwriting of the output files.
```

2.16 scil_compute_MT_maps.py

```
usage: __main__.py [-h] [--out_prefix OUT_PREFIX] [--in_B1_map IN_B1_MAP]
                   [--in_mtoff IN_MTOFF [IN_MTOFF ...]]
                   [--in_mton IN_MTON [IN_MTON ...]]
                   [--in_t1w IN_T1W [IN_T1W ...]] [-f]
                   out_dir in_mask

This script computes two myelin indices maps from the Magnetization Transfer
(MT) images.
Magnetization Transfer is a contrast mechanism in tissue resulting from the
proton exchange between non-aqueous protons (from macromolecules and their
closely associated water molecules, the "bound" pool) and protons in the free
water pool called aqueous protons. This exchange attenuates the MRI signal,
introducing microstructure-dependent contrast. MT's effect reflects the
relative density of macromolecules such as proteins and lipids, it has been
associated with myelin content in white matter of the brain.

Different contrasts can be done with an off-resonance pulse to saturating the
protons on non-aqueous molecules a frequency irradiation. The MT maps are
obtained using three contrasts: single frequency irradiation (MT-on, saturated
images) and an unsaturated contrast (MT-off); and a T1weighted image as
reference.

The output consist in two types of images:
        Three contrasts images : MT-off, MT-on and T1weighted images.
        MT maps corrected or not for an empiric B1 correction maps.

Input Data recommendation:
  - it is recommended to use dcm2niix (v1.0.20200331) to convert data
    https://github.com/rordenlab/dcm2niix/releases/tag/v1.0.20200331
  - dcm2niix conversion will create all echo files for each contrast and
    corresponding json files
  - all input must have a matching json file with the same filename
  - all contrasts must have a same number of echoes and coregistered
    between them before running the script.
```

(continues on next page)

(continued from previous page)

```
  - Mask must be coregistered to the echo images
  - ANTs can be used for the registration steps (http://stnava.github.io/ANTs/)

The output consist in two types of images in two folders :
  1. Contrasts_MT_maps which contains the 2 contrast images
      - MT-off.nii.gz : pulses applied at positive frequency
      - MT-on.nii.gz : pulses applied at negative frequency
      - T1w.nii.gz : anatomical T1 reference images

  2. MT_native_maps which contains the 4 myelin maps
      - MTR.nii.gz : Magnetization Transfer Ratio map
      The MT ratio is a measure reflecting the amount of bound protons.

      - MTsat.nii.gz : Magnetization Transfer saturation map
      The MT saturation is a pseudo-quantitative maps representing
      the signal change between the bound and free water pools.

>>> scil_compute_ihMT_maps.py path/to/output/directory path/to/mask_bin.nii.gz
    --in_mtoff path/to/echo*mtoff.nii.gz --in_mton path/to/echo*mton.nii.gz
    --in_t1w path/to/echo*T1w.nii.gz

positional arguments:
  out_dir               Path to output folder.
  in_mask               Path to the T1 binary brain mask. Must be the sum of the␣
↪three tissue probability maps from T1 segmentation (GM+WM+CSF).

optional arguments:
  -h, --help            show this help message and exit
  --out_prefix OUT_PREFIX
                        Prefix to be used for each output image.
  --in_B1_map IN_B1_MAP
                        Path to B1 coregister map to MT contrasts.
  -f                    Force overwriting of the output files.

MT contrasts:
  Path to echoes corresponding to contrasts images. All constrasts must have the same␣
↪number of echoes and coregistered between them.Use * to include all echoes.

  --in_mtoff IN_MTOFF [IN_MTOFF ...]
                        Path to all echoes corresponding to the no frequency␣
↪saturation pulse (reference image).
  --in_mton IN_MTON [IN_MTON ...]
                        Path to all echoes corresponding to the Positive frequency␣
↪saturation pulse.
  --in_t1w IN_T1W [IN_T1W ...]
                        Path to all echoes corresponding to the T1-weigthed.
```

2.17 scil_compute_NODDI.py

```
usage: __main__.py [-h] [--mask MASK] [--out_dir OUT_DIR] [--b_thr B_THR]
                   [--para_diff PARA_DIFF] [--iso_diff ISO_DIFF]
                   [--lambda1 LAMBDA1] [--lambda2 LAMBDA2]
                   [--save_kernels DIRECTORY | --load_kernels DIRECTORY]
                   [--compute_only] [--processes NBR] [-f] [-v]
```

(continues on next page)

(continued from previous page)

```
                    in_dwi in_bval in_bvec

Compute NODDI [1] maps using AMICO.
Multi-shell DWI necessary.

positional arguments:
  in_dwi                DWI file acquired with a NODDI compatible protocol
                        (single-shell data not suited).
  in_bval               b-values filename, in FSL format (.bval).
  in_bvec               b-vectors filename, in FSL format (.bvec).

optional arguments:
  -h, --help            show this help message and exit
  --mask MASK           Brain mask filename.
  --out_dir OUT_DIR     Output directory for the NODDI results. [results]
  --b_thr B_THR         Limit value to consider that a b-value is on an
                        existing shell. Above this limit, the b-value is
                        placed on a new shell. This includes b0s values.
  --processes NBR       Number of sub-processes to start. Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Model options:
  --para_diff PARA_DIFF
                        Axial diffusivity (AD) in the CC. [0.0017]
  --iso_diff ISO_DIFF   Mean diffusivity (MD) in ventricles. [0.003]
  --lambda1 LAMBDA1     First regularization parameter. [0.5]
  --lambda2 LAMBDA2     Second regularization parameter. [0.001]

Kernels options:
  --save_kernels DIRECTORY
                        Output directory for the COMMIT kernels.
  --load_kernels DIRECTORY
                        Input directory where the COMMIT kernels are located.
  --compute_only        Compute kernels only, --save_kernels must be used.

Reference:
    [1] Zhang H, Schneider T, Wheeler-Kingshott CA, Alexander DC.
        NODDI: practical in vivo neurite orientation dispersion
        and density imaging of the human brain.
        NeuroImage. 2012 Jul 16;61:1000-16.
```

2.18 scil_compute_NODDI_priors.py

```
usage: __main__.py [-h] [--fa_min FA_MIN] [--fa_max FA_MAX] [--md_min MD_MIN]
                   [--roi_radius ROI_RADIUS]
                   [--roi_center tuple3) [tuple(3 ...]]
                   [--out_txt_1fiber FILE] [--out_mask_1fiber FILE]
                   [--out_txt_ventricles FILE] [--out_mask_ventricles FILE]
                   [-f] [-v]
                   in_FA in_AD in_MD

Compute the axial (para_diff) and mean (iso_diff) diffusivity priors for NODDI.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_FA                 Path to the FA volume.
  in_AD                 Path to the axial diffusivity (AD) volume.
  in_MD                 Path to the mean diffusivity (MD) volume.

optional arguments:
  -h, --help            show this help message and exit
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Metrics options:
  --fa_min FA_MIN       Minimal threshold of FA (voxels above that threshold
                        are considered in the single fiber mask). [0.7]
  --fa_max FA_MAX       Maximal threshold of FA (voxels under that threshold
                        are considered in the ventricles). [0.1]
  --md_min MD_MIN       Minimal threshold of MD in mm2/s (voxels above that
                        threshold are considered for in the ventricles).
                        [0.003]

Regions options:
  --roi_radius ROI_RADIUS
                        Radius of the region used to estimate the priors. The
                        roi will be a cube spanning from ROI_CENTER in each
                        direction. [20]
  --roi_center tuple(3) [tuple(3) ...]
                        Center of the roi of size roi_radius used to estimate
                        the priors. [center of the 3D volume]

Outputs:
  --out_txt_1fiber FILE
                        Output path for the text file containing the single
                        fiber average value of AD. If not set, the file will
                        not be saved.
  --out_mask_1fiber FILE
                        Output path for single fiber mask. If not set, the
                        mask will not be saved.
  --out_txt_ventricles FILE
                        Output path for the text file containing the
                        ventricles average value of MD. If not set, the file
                        will not be saved.
  --out_mask_ventricles FILE
                        Output path for the ventricule mask. If not set, the
                        mask will not be saved.

Reference:
    [1] Zhang H, Schneider T, Wheeler-Kingshott CA, Alexander DC.
        NODDI: practical in vivo neurite orientation dispersion
        and density imaging of the human brain.
        NeuroImage. 2012 Jul 16;61:1000-16.
```

2.19 scil_compute_asym_odf_metrics.py

```
usage: __main__.py [-h] [--mask MASK] [--cos_asym_map COS_ASYM_MAP]
                   [--odd_power_map ODD_POWER_MAP] [--peaks PEAKS]
```

(continues on next page)

(continued from previous page)

```
                    [--peak_values PEAK_VALUES] [--peak_indices PEAK_INDICES]
                    [--nupeaks NUPEAKS] [--not_all] [--at A_THRESHOLD]
                    [--rt R_THRESHOLD]
                    [--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724}]
                    [--processes NBR] [--sh_basis {descoteaux07,tournier07}]
                    [-f]
                    in_sh

Script to compute various metrics derivated from asymmetric ODF.

These metrics include an asymmetric peak directions image, a number of peaks
(nupeaks) map [2], a cosine-similarity-based asymmetry map [1] and an
odd-power map [2].

The asymmetric peak directions image contains peaks per hemisphere, considering
antipodal sphere directions as distinct. On a symmetric signal, the number of
asymmetric peaks extracted is then twice the number of symmetric peaks.

The nupeaks map is the asymmetric alternative to NuFO maps. It counts the
number of asymmetric peaks extracted and ranges in [0..N] with N the maximum
number of peaks.

The cosine-based asymmetry map is in the range [0..1], with 0 corresponding
to a perfectly symmetric signal and 1 to a perfectly asymmetric signal.

The odd-power map is also in the range [0..1], with 0 corresponding
to a perfectly symmetric signal and 1 to a perfectly anti-symmetric signal. It
is given by the ratio of the L2-norm of odd SH coefficients on the L2-norm of
all SH coefficients.

positional arguments:
  in_sh                 Input SH image.

optional arguments:
  -h, --help            show this help message and exit
  --mask MASK           Optional mask.
  --cos_asym_map COS_ASYM_MAP
                        Output asymmetry map using cos similarity.
  --odd_power_map ODD_POWER_MAP
                        Output odd power map.
  --peaks PEAKS         Output filename for the extracted peaks.
  --peak_values PEAK_VALUES
                        Output filename for the extracted peaks values.
  --peak_indices PEAK_INDICES
                        Output filename for the generated peaks indices on the sphere.
  --nupeaks NUPEAKS     Output filename for the nupeaks file.
  --not_all             If set, only saves the files specified using the file flags␣
↪[False].
  --at A_THRESHOLD      Absolute threshold on fODF amplitude. This value should be␣
↪set to
                        approximately 1.5 to 2 times the maximum fODF amplitude in␣
↪isotropic voxels
                        (ie. ventricles).
                        Use compute_fodf_max_in_ventricles.py to find the maximal␣
↪value.
                        See [Dell'Acqua et al HBM 2013] [0.0].
```

(continues on next page)

(continued from previous page)

```
  --rt R_THRESHOLD      Relative threshold on fODF amplitude in percentage [0.1].
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Sphere to use for peak directions estimation [symmetric724].
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -f                    Force overwriting of the output files.

References:
[1] S. Cetin Karayumak, E. Özarslan, and G. Unal,
"Asymmetric Orientation Distribution Functions (AODFs) revealing
intravoxel geometry in diffusion MRI," Magnetic Resonance Imaging,
vol. 49, pp. 145-158, Jun. 2018, doi: 10.1016/j.mri.2018.03.006.

[2] C. Poirier, E. St-Onge, and M. Descoteaux, "Investigating the Occurence of
Asymmetric Patterns in White Matter Fiber Orientation Distribution Functions"
[Abstract], In: Proc. Intl. Soc. Mag. Reson. Med. 29 (2021), 2021 May 15-20,
Vancouver, BC, Abstract number 0865.
```

2.20 scil_compute_bundle_mean_std.py

```
usage: __main__.py [-h] [--density_weighting]
                   [--distance_weighting DISTANCE_NII]
                   [--correlation_weighting CORRELATION_NII] [--include_dps]
                   [--reference REFERENCE] [--indent INDENT] [--sort_keys]
                   in_bundle in_metrics [in_metrics ...]

Compute mean and std for the whole bundle for each metric. This is achieved by
averaging the metrics value of all voxels occupied by the bundle.

Density weighting modifies the contribution of voxel with lower/higher
streamline count to reduce influence of spurious streamlines.

positional arguments:
  in_bundle             Fiber bundle file to compute statistics on
  in_metrics            Nifti file to compute statistics on. Probably some␣
↪tractometry measure(s) such as FA, MD, RD, ...

optional arguments:
  -h, --help            show this help message and exit
  --density_weighting   If set, weight statistics by the number of fibers passing␣
↪through each voxel.
  --distance_weighting DISTANCE_NII
                        If set, weight statistics by the inverse of the distance␣
↪between a streamline and the centroid.
  --correlation_weighting CORRELATION_NII
                        If set, weight statistics by the correlation strength between␣
↪longitudinal data.
```

(continues on next page)

(continued from previous page)

```
  --include_dps         Save values from data_per_streamline.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.21 scil_compute_bundle_mean_std_per_point.py

```
usage: __main__.py [-h] [--density_weighting]
                   [--distance_weighting DISTANCE_NII]
                   [--correlation_weighting CORRELATION_NII]
                   [--out_json OUT_JSON] [-f] [--reference REFERENCE]
                   [--indent INDENT] [--sort_keys]
                   in_bundle in_labels in_metrics [in_metrics ...]

Compute mean and standard deviation for all streamlines points in the bundle
for each metric combination, along the bundle, i.e. for each point.

**To create label_map and distance_map, see scil_label_and_distance_maps.py.

positional arguments:
  in_bundle             Fiber bundle file to compute statistics on.
  in_labels             Label map (.nii.gz) of the corresponding fiber bundle.
  in_metrics            Nifti file to compute statistics on. Probably some␣
↪tractometry measure(s) such as FA, MD, RD, ...

optional arguments:
  -h, --help            show this help message and exit
  --density_weighting   If set, weight statistics by the number of streamlines␣
↪passing through each voxel.
  --distance_weighting DISTANCE_NII
                        If set, weight statistics by the inverse of the distance␣
↪between a streamline and the centroid.
  --correlation_weighting CORRELATION_NII
                        If set, weight statistics by the correlation strength between␣
↪longitudinal data.
  --out_json OUT_JSON   Path of the output json file. If not given, json formatted␣
↪stats are simply printed.
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.22 scil_compute_bundle_volume.py

```
usage: __main__.py [-h] [--reference REFERENCE] [--indent INDENT]
                   [--sort_keys]
                   in_bundle

Compute bundle volume in mm3. This script supports anisotropic voxels
resolution. Volume is estimated by counting the number of voxels occupied by
the bundle and multiplying it by the volume of a single voxel.

This estimation is typically performed at resolution around 1mm3.

positional arguments:
  in_bundle             Fiber bundle file.

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.23 scil_compute_bundle_volume_per_label.py

```
usage: __main__.py [-h] [--indent INDENT] [--sort_keys] [-f]
                   voxel_label_map bundle_name

Compute bundle volume per label in mm3. This script supports anisotropic voxels
resolution. Volume is estimated by counting the number of voxel occupied by
each label and multiplying it by the volume of a single voxel.

This estimation is typically performed at resolution around 1mm3.

positional arguments:
  voxel_label_map  Fiber bundle file.
  bundle_name      Bundle name.

optional arguments:
  -h, --help       show this help message and exit
  -f               Force overwriting of the output files.

Json options:
  --indent INDENT  Indent for json pretty print.
  --sort_keys      Sort keys in output json.
```

2.24 scil_compute_bundle_voxel_label_map.py

```
usage: __main__.py [-h] [--nb_pts NB_PTS] [--new_labeling]
                   [--min_streamline_count MIN_STREAMLINE_COUNT]
                   [--min_voxel_count MIN_VOXEL_COUNT] [--colormap COLORMAP]
                   [--reference REFERENCE] [-f]
                   in_bundles [in_bundles ...] in_centroid out_dir

Compute label image (Nifti) from bundle and centroid.
Each voxel will have the label of its nearest centroid point.

The number of labels will be the same as the centroid's number of points.

positional arguments:
  in_bundles            Fiber bundle file.
  in_centroid           Centroid streamline corresponding to bundle.
  out_dir               Directory to save all mapping and coloring file.

optional arguments:
  -h, --help            show this help message and exit
  --nb_pts NB_PTS       Number of divisions for the bundles.
                        Default is the number of points of the centroid.
  --new_labeling        Activate the new labeling method based on clusters.
  --min_streamline_count MIN_STREAMLINE_COUNT
                        Minimum number of streamlines for filtering/cuttingoperation␣
↪[100000].
  --min_voxel_count MIN_VOXEL_COUNT
                        Minimum number of voxels for filtering/cuttingoperation␣
↪[1000000].
  --colormap COLORMAP   Select the colormap for colored trk (data_per_point) [jet].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.25 scil_compute_centroid.py

```
usage: __main__.py [-h] [--nb_points NB_POINTS] [--reference REFERENCE] [-f]
                   in_bundle out_centroid

Compute a single bundle centroid, using an 'infinite' QuickBundles threshold.

positional arguments:
  in_bundle             Fiber bundle file.
  out_centroid          Output centroid streamline filename.

optional arguments:
  -h, --help            show this help message and exit
  --nb_points NB_POINTS
                        Number of points defining the centroid streamline[20].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.26 scil_compute_connectivity.py

```
usage: __main__.py [-h] [--volume OUT_FILE] [--streamline_count OUT_FILE]
                   [--length OUT_FILE] [--similarity IN_FOLDER OUT_FILE]
                   [--maps IN_FOLDER OUT_FILE] [--metrics IN_FILE OUT_FILE]
                   [--lesion_load IN_FILE OUT_DIR]
                   [--min_lesion_vol MIN_LESION_VOL] [--density_weighting]
                   [--no_self_connection] [--include_dps OUT_DIR]
                   [--force_labels_list FORCE_LABELS_LIST] [--processes NBR]
                   [-v] [-f]
                   in_hdf5 in_labels

This script computes a variety of measures in the form of connectivity
matrices. This script is made to follow scil_decompose_connectivity and
uses the same labels list as input.

The script expects a folder containing all relevants bundles following the
naming convention LABEL1_LABEL2.trk and a text file containing the list of
labels that should be part of the matrices. The ordering of labels in the
matrices will follow the same order as the list.
This script only generates matrices in the form of array, does not visualize
or reorder the labels (node).

The parameter --similarity expects a folder with density maps
(LABEL1_LABEL2.nii.gz) following the same naming convention as the input
directory.
The bundles should be averaged version in the same space. This will
compute the weighted-dice between each node and their homologuous average
version.

The parameters --metrics can be used more than once and expect a map (t1, fa,
etc.) in the same space and each will generate a matrix. The average value in
the volume occupied by the bundle will be reported in the matrices nodes.

The parameters --maps can be used more than once and expect a folder with
pre-computed maps (LABEL1_LABEL2.nii.gz) following the same naming convention
as the input directory. Each will generate a matrix. The average non-zeros
value in the map will be reported in the matrices nodes.

The parameters --lesion_load will compute 3 lesion(s) related matrices:
lesion_count.npy, lesion_vol.npy, lesion_sc.npy and put it inside of a
specified folder. They represent the number of lesion, the total volume of
lesion(s) and the total of streamlines going through the lesion(s) for  of each
connection. Each connection can be seen as a 'bundle' and then something
similar to scil_analyse_lesion_load.py is run for each 'bundle'.

positional arguments:
  in_hdf5               Input filename for the hdf5 container (.h5).
                        Obtained from scil_decompose_connectivity.py.
  in_labels             Labels file name (nifti).
                        This generates a NxN connectivity matrix.

optional arguments:
  -h, --help            show this help message and exit
  --volume OUT_FILE     Output file for the volume weighted matrix (.npy).
  --streamline_count OUT_FILE
                        Output file for the streamline count weighted matrix (.npy).
```

(continues on next page)

(continued from previous page)

```
  --length OUT_FILE     Output file for the length weighted matrix (.npy).
  --similarity IN_FOLDER OUT_FILE
                        Input folder containing the averaged bundle density
                        maps (.nii.gz) and output file for the similarity weighted␣
↪matrix (.npy).
  --maps IN_FOLDER OUT_FILE
                        Input folder containing pre-computed maps (.nii.gz)
                        and output file for the weighted matrix (.npy).
  --metrics IN_FILE OUT_FILE
                        Input (.nii.gz). and output file (.npy) for a metric weighted␣
↪matrix.
  --lesion_load IN_FILE OUT_DIR
                        Input binary mask (.nii.gz) and output directory for all␣
↪lesion-related matrices.
  --min_lesion_vol MIN_LESION_VOL
                        Minimum lesion volume in mm3 [7].
  --density_weighting   Use density-weighting for the metric weighted matrix.
  --no_self_connection  Eliminate the diagonal from the matrices.
  --include_dps OUT_DIR
                        Save matrices from data_per_streamline in the output␣
↪directory.
                        COMMIT-related values will be summed instead of averaged.
                        Will always overwrite files.
  --force_labels_list FORCE_LABELS_LIST
                        Path to a labels list (.txt) in case of missing labels in the␣
↪atlas.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.27 scil_compute_divide.py

```
usage: __main__.py [-h] --in_dwis IN_DWIS [IN_DWIS ...] --in_bvals IN_BVALS
                   [IN_BVALS ...] --in_bvecs IN_BVECS [IN_BVECS ...]
                   --in_bdeltas {0,1,-0.5,0.5} [{0,1,-0.5,0.5} ...]
                   [--mask MASK] [--fa FA] [--tolerance TOLERANCE]
                   [--fit_iters FIT_ITERS] [--random_iters RANDOM_ITERS]
                   [--do_weight_bvals] [--do_weight_pa] [--do_multiple_s0]
                   [--force_b0_threshold] [--processes NBR] [-f] [-v]
                   [--not_all] [--md file] [--ufa file] [--op file]
                   [--mk_i file] [--mk_a file] [--mk_t file]

Script to compute microstructure metrics using the DIVIDE method. In order to
operate, the script needs at leats two different types of b-tensor encodings.
Note that custom encodings are not yet supported, so that only the linear
tensor encoding (LTE, b_delta = 1), the planar tensor encoding
(PTE, b_delta = -0.5), the spherical tensor encoding (STE, b_delta = 0) and
the cigar shape tensor encoding (b_delta = 0.5) are available. Moreover, all
of `--in_dwis`, `--in_bvals`, `--in_bvecs` and `--in_bdeltas` must have the
same number of arguments. Be sure to keep the same order of encodings
throughout all these inputs and to set `--in_bdeltas` accordingly (IMPORTANT).

By default, will output all possible files, using default names.
```

(continues on next page)

(continued from previous page)

```
Specific names can be specified using the file flags specified in the
"File flags" section.

If --not_all is set, only the files specified explicitly by the flags
will be output.

Based on Markus Nilsson, Filip Szczepankiewicz, Björn Lampinen, André Ahlgren,
João P. de Almeida Martins, Samo Lasic, Carl-Fredrik Westin,
and Daniel Topgaard. An open-source framework for analysis of multidimensional
diffusion MRI data implemented in MATLAB.
Proc. Intl. Soc. Mag. Reson. Med. (26), Paris, France, 2018.

optional arguments:
  -h, --help            show this help message and exit
  --in_dwis IN_DWIS [IN_DWIS ...]
                        Path to the input diffusion volume for each b-tensor encoding␣
↪type.
  --in_bvals IN_BVALS [IN_BVALS ...]
                        Path to the bval file, in FSL format, for each b-tensor␣
↪encoding type.
  --in_bvecs IN_BVECS [IN_BVECS ...]
                        Path to the bvec file, in FSL format, for each b-tensor␣
↪encoding type.
  --in_bdeltas {0,1,-0.5,0.5} [{0,1,-0.5,0.5} ...]
                        Value of b_delta for each b-tensor encoding type, in the same␣
↪order as dwi, bval and bvec inputs.
  --mask MASK           Path to a binary mask. Only the data inside the mask will be␣
↪used for computations and reconstruction.
  --fa FA               Path to a FA map. Needed for calculating the OP.
  --tolerance TOLERANCE
                        The tolerated gap between the b-values to extract
                        and the current b-value. [20]
  --fit_iters FIT_ITERS
                        The number of time the gamma fit will be done [1]
  --random_iters RANDOM_ITERS
                        The number of iterations for the initial parameters search.␣
↪[50]
  --do_weight_bvals     If set, does not do a weighting on the bvalues in the gamma␣
↪fit.
  --do_weight_pa        If set, does not do a powder averaging weighting in the gamma␣
↪fit.
  --do_multiple_s0      If set, does not take into account multiple baseline signals.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --not_all             If set, only saves the files specified using the file flags.␣
↪(Default: False)

File flags:
  --md file             Output filename for the MD.
  --ufa file            Output filename for the microscopic FA.
  --op file             Output filename for the order parameter.
  --mk_i file           Output filename for the isotropic mean kurtosis.
  --mk_a file           Output filename for the anisotropic mean kurtosis.
```

(continues on next page)

(continued from previous page)

```
  --mk_t file           Output filename for the total mean kurtosis.
```

2.28 scil_compute_dti_metrics.py

```
usage: __main__.py [-h] [-f] [--mask MASK] [--method method_name] [--not_all]
                   [--ad file] [--evecs file] [--evals file] [--fa file]
                   [--ga file] [--md file] [--mode file] [--norm file]
                   [--rgb file] [--rd file] [--tensor file]
                   [--tensor_format {fsl,nifti,mrtrix,dipy}]
                   [--non-physical file] [--pulsation string]
                   [--residual file] [--force_b0_threshold]
                   in_dwi in_bval in_bvec

Script to compute all of the Diffusion Tensor Imaging (DTI) metrics.

By default, will output all available metrics, using default names. Specific
names can be specified using the metrics flags that are listed in the "Metrics
files flags" section.

If --not_all is set, only the metrics specified explicitly by the flags
will be output. The available metrics are:

fractional anisotropy (FA), geodesic anisotropy (GA), axial diffusivisty (AD),
radial diffusivity (RD), mean diffusivity (MD), mode, red-green-blue colored
FA (rgb), principal tensor e-vector and tensor coefficients (dxx, dxy, dxz,
dyy, dyz, dzz).

For all the quality control metrics such as residual, physically implausible
signals, pulsation and misalignment artifacts, see
[J-D Tournier, S. Mori, A. Leemans. Diffusion Tensor Imaging and Beyond.
MRM 2011].

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  in_bvec               Path of the bvecs file, in FSL format.

optional arguments:
  -h, --help            show this help message and exit
  -f                    Force overwriting of the output files.
  --mask MASK           Path to a binary mask.
                        Only data inside the mask will be used for computations and␣
↪reconstruction. (Default: None)
  --method method_name  Tensor fit method.
                        WLS for weighted least squares
                        LS for ordinary least squares
                        NLLS for non-linear least-squares
                        restore for RESTORE robust tensor fitting. (Default: WLS)
  --not_all             If set, will only save the metrics explicitly specified using␣
↪the other metrics flags. (Default: not set).
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)

Metrics files flags:
```

(continues on next page)

(continued from previous page)

```
  --ad file             Output filename for the axial diffusivity.
  --evecs file          Output filename for the eigenvectors of the tensor.
  --evals file          Output filename for the eigenvalues of the tensor.
  --fa file             Output filename for the fractional anisotropy.
  --ga file             Output filename for the geodesic anisotropy.
  --md file             Output filename for the mean diffusivity.
  --mode file           Output filename for the mode.
  --norm file           Output filename for the tensor norm.
  --rgb file            Output filename for the colored fractional anisotropy.
  --rd file             Output filename for the radial diffusivity.
  --tensor file         Output filename for the tensor coefficients.
  --tensor_format {fsl,nifti,mrtrix,dipy}
                        Format used for the tensors saved in --tensor file.(default:␣
↪fsl)

                            Dipy's order is [Dxx, Dxy, Dyy, Dxz, Dyz, Dzz]
                               Shape: [i, j , k, 6].
                               Ref: https://github.com/dipy/dipy/blob/master/dipy/
↪reconst/dti.py#L1639

                            MRTRIX's order is : [Dxx, Dyy, Dzz, Dxy, Dxz, Dyz]
                               Shape: [i, j , k, 6].
                               Ref: https://mrtrix.readthedocs.io/en/dev/reference/
↪commands/dwi2tensor.html

                            ANTS's order ('nifti format') is : [Dxx, Dxy, Dyy, Dxz,␣
↪Dyz, Dzz].
                               Shape: [i, j , k, 1, 6] (Careful, file is 5D).
                               Ref: https://github.com/ANTsX/ANTs/wiki/Importing-
↪diffusion-tensor-data-from-other-software

                            FSL's order is [Dxx, Dxy, Dxz, Dyy, Dyz, Dzz]
                               Shape: [i, j , k, 6].
                               Ref: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FDT/
↪UserGuide
                               (Also used for the Fibernavigator)


Quality control files flags:
  --non-physical file   Output filename for the voxels with physically implausible␣
↪signals
                        where the mean of b=0 images is below one or more diffusion-
↪weighted images.
  --pulsation string    Standard deviation map across all diffusion-weighted images␣
↪and across b=0 images if more than one is available.
                        Shows pulsation and misalignment artifacts.
  --residual file       Output filename for the map of the residual of the tensor fit.
```

2.29 scil_compute_endpoints_map.py

```
usage: __main__.py [-h] [--swap] [--binary] [--indent INDENT] [--sort_keys]
                   [--reference REFERENCE] [-f]
                   in_bundle endpoints_map_head endpoints_map_tail
```

(continues on next page)

(continued from previous page)

```
Computes the endpoint map of a bundle. The endpoint map is simply a count of
the number of streamlines that start or end in each voxel.

The idea is to estimate the cortical area affected by the bundle (assuming
streamlines start/end in the cortex).

Note: If the streamlines are not ordered the head/tail are random and not
really two coherent groups. Use the following script to order streamlines:
scil_uniformize_streamlines_endpoints.py

positional arguments:
  in_bundle             Fiber bundle filename.
  endpoints_map_head    Output endpoints map head filename.
  endpoints_map_tail    Output endpoints map tail filename.

optional arguments:
  -h, --help            show this help message and exit
  --swap                Swap head<->tail convention. Can be useful when the reference␣
↪is not in RAS.
  --binary              Save outputs as a binary mask instead of a heat map.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.30 scil_compute_fodf_max_in_ventricles.py

```
usage: __main__.py [-h] [--fa_t FA_THRESHOLD] [--md_t MD_THRESHOLD]
                   [--max_value_output file] [--mask_output file]
                   [--small_dims] [--sh_basis {descoteaux07,tournier07}] [-v]
                   [-f]
                   fODFs FA MD

Script to compute the maximum fODF in the ventricles. The ventricules are
estimated from a MD and FA threshold.

This allows to clip the noise of fODF using an absolute thresold.

positional arguments:
  fODFs                 Path of the fODF volume in spherical harmonics (SH).
  FA                    Path to the FA volume.
  MD                    Path to the mean diffusivity (MD) volume.

optional arguments:
  -h, --help            show this help message and exit
  --fa_t FA_THRESHOLD   Maximal threshold of FA (voxels under that threshold are␣
↪considered for evaluation, [0.1]).
  --md_t MD_THRESHOLD   Minimal threshold of MD in mm2/s (voxels above that threshold␣
↪are considered for evaluation, [0.003]).
  --max_value_output file
```

(continues on next page)

(continued from previous page)

```
                        Output path for the text file containing the value. If not␣
↪set the file will not be saved.
  --mask_output file    Output path for the ventricule mask. If not set, the mask␣
↪will not be saved.
  --small_dims          If set, takes the full range of data to search the max fodf␣
↪amplitude in ventricles. Useful when the data is 2D or has small dimensions.
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

[1] Dell'Acqua, Flavio, et al. "Can spherical deconvolution provide more
    information than fiber orientations? Hindrance modulated orientational
    anisotropy, a true-tract specific index to characterize white matter
    diffusion." Human brain mapping 34.10 (2013): 2464-2483.
```

2.31 scil_compute_fodf_metrics.py

```
usage: __main__.py [-h] [--sphere string] [--mask] [--at A_THRESHOLD]
                   [--rt R_THRESHOLD] [--sh_basis {descoteaux07,tournier07}]
                   [-f] [--processes NBR] [--not_all] [--afd_max file]
                   [--afd_total file] [--afd_sum file] [--nufo file]
                   [--rgb file] [--peaks file] [--peak_values file]
                   [--peak_indices file]
                   in_fODF

Script to compute the maximum Apparent Fiber Density (AFD), the fiber ODFs
orientations, values and indices (peaks, peak_values, peak_indices), the Number
of Fiber Orientations (NuFO) maps from fiber ODFs and the RGB map.

AFD_max map is the maximal fODF amplitude for each voxel.

NuFO is the the number of maxima of the fODF with an ABSOLUTE amplitude above
the threshold set using --at, AND an amplitude above the RELATIVE threshold
set using --rt.

The --at argument should be set to a value which is 1.5 times the maximal
value of the fODF in the ventricules. This can be obtained with the
compute_fodf_max_in_ventricules.py script.

By default, will output all possible files, using default names. Specific names
can be specified using the file flags specified in the "File flags" section.

If --not_all is set, only the files specified explicitly by the flags will be
output.

See [Raffelt et al. NeuroImage 2012] and [Dell'Acqua et al HBM 2013] for the
definitions.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_fODF               Path of the fODF volume in spherical harmonics (SH).

optional arguments:
  -h, --help            show this help message and exit
  --sphere string       Discrete sphere to use in the processing [repulsion724].
  --mask                Path to a binary mask. Only the data inside the mask
                        will beused for computations and reconstruction [None].
  --at A_THRESHOLD      Absolute threshold on fODF amplitude. This value should be␣
↪set to
                        approximately 1.5 to 2 times the maximum fODF amplitude in␣
↪isotropic voxels
                        (ie. ventricles).
                        Use compute_fodf_max_in_ventricles.py to find the maximal␣
↪value.
                        See [Dell'Acqua et al HBM 2013] [0.0].
  --rt R_THRESHOLD      Relative threshold on fODF amplitude in percentage  [0.1].
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -f                    Force overwriting of the output files.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --not_all             If set, only saves the files specified using the file flags␣
↪[False].

File flags:
  --afd_max file        Output filename for the AFD_max map.
  --afd_total file      Output filename for the AFD_total map (SH coeff = 0).
  --afd_sum file        Output filename for the sum of all peak contributions
                        (sum of fODF lobes on the sphere).
  --nufo file           Output filename for the NuFO map.
  --rgb file            Output filename for the RGB map.
  --peaks file          Output filename for the extracted peaks.
  --peak_values file    Output filename for the extracted peaks values.
  --peak_indices file   Output filename for the generated peaks indices on the sphere.
```

2.32 scil_compute_freewater.py

```
usage: __main__.py [-h] [--mask MASK] [--out_dir OUT_DIR] [--b_thr B_THR]
                   [--para_diff PARA_DIFF] [--iso_diff ISO_DIFF]
                   [--perp_diff_min PERP_DIFF_MIN]
                   [--perp_diff_max PERP_DIFF_MAX] [--lambda1 LAMBDA1]
                   [--lambda2 LAMBDA2]
                   [--save_kernels DIRECTORY | --load_kernels DIRECTORY]
                   [--compute_only] [--mouse] [--processes NBR] [-f] [-v]
                   in_dwi in_bval in_bvec

Compute Free Water maps [1] using AMICO.
This script supports both single and multi-shell data.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_dwi                DWI file.
  in_bval               b-values filename, in FSL format (.bval).
  in_bvec               b-vectors filename, in FSL format (.bvec).

optional arguments:
  -h, --help            show this help message and exit
  --mask MASK           Brain mask filename.
  --out_dir OUT_DIR     Output directory for the Free Water results. [results]
  --b_thr B_THR         Limit value to consider that a b-value is on an
                        existing shell. Above this limit, the b-value is
                        placed on a new shell. This includes b0s values.
  --mouse               If set, use mouse fitting profile.
  --processes NBR       Number of sub-processes to start. Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Model options:
  --para_diff PARA_DIFF
                        Axial diffusivity (AD) in the CC. [0.0015]
  --iso_diff ISO_DIFF   Mean diffusivity (MD) in ventricles. [0.003]
  --perp_diff_min PERP_DIFF_MIN
                        Radial diffusivity (RD) minimum. [0.0001]
  --perp_diff_max PERP_DIFF_MAX
                        Radial diffusivity (RD) maximum. [0.0007]
  --lambda1 LAMBDA1     First regularization parameter. [0.0]
  --lambda2 LAMBDA2     Second regularization parameter. [0.25]

Kernels options:
  --save_kernels DIRECTORY
                        Output directory for the COMMIT kernels.
  --load_kernels DIRECTORY
                        Input directory where the COMMIT kernels are located.
  --compute_only        Compute kernels only, --save_kernels must be used.

Reference:
    [1] Pasternak 0, Sochen N, Gur Y, Intrator N, Assaf Y.
        Free water elimination and mapping from diffusion mri.
        Magn Reson Med. 62 (3) (2009) 717-730.
```

2.33 scil_compute_hdf5_average_density_map.py

```
usage: __main__.py [-h] [--binary] [--processes NBR] [-f]
                   in_hdf5 [in_hdf5 ...] out_dir

Compute a density map for each connection from a hdf5 file.
Typically use after scil_decompose_connectivity.py in order to obtain the
average density map of each connection to allow the use of --similarity
in scil_compute_connectivity.py.

This script is parallelized, but will run much slower on non-SSD if too many
processes are used. The output is a directory containing the thousands of
connections:
```

(continues on next page)

(continued from previous page)

```
out_dir/
    |-- LABEL1_LABEL1.nii.gz
    |-- LABEL1_LABEL2.nii.gz
    |-- [...]
    |-- LABEL90_LABEL90.nii.gz

positional arguments:
  in_hdf5           List of HDF5 filenames (.h5) from scil_decompose_connectivity.py.
  out_dir           Path of the output directory.

optional arguments:
  -h, --help        show this help message and exit
  --binary          Binarize density maps before the population average.
  --processes NBR   Number of sub-processes to start.
                    Default: [1]
  -f                Force overwriting of the output files.
```

2.34 scil_compute_ihMT_maps.py

```
usage: __main__.py [-h] [--out_prefix OUT_PREFIX] [--in_B1_map IN_B1_MAP]
                   [--filtering] [--single_echo] --in_altnp IN_ALTNP
                   [IN_ALTNP ...] --in_altpn IN_ALTPN [IN_ALTPN ...]
                   --in_mtoff IN_MTOFF [IN_MTOFF ...] --in_negative
                   IN_NEGATIVE [IN_NEGATIVE ...] --in_positive IN_POSITIVE
                   [IN_POSITIVE ...] --in_t1w IN_T1W [IN_T1W ...] [-f]
                   out_dir in_mask

This script computes four myelin indices maps from the Magnetization Transfer
(MT) and inhomogeneous Magnetization Transfer (ihMT) images. Magnetization
Transfer is a contrast mechanism in tissue resulting from the proton exchange
between non-aqueous protons (from macromolecules and their closely associated
water molecules, the "bound" pool) and protons in the free water pool called
aqueous protons. This exchange attenuates the MRI signal, introducing
microstructure-dependent contrast. MT's effect reflects the relative density
of macromolecules such as proteins and lipids, it has been associated with
myelin content in white matter of the brain.

Different contrasts can be done with an off-resonance pulse prior to image
acquisition (a prepulse), saturating the protons on non-aqueous molecules,
by applying different frequency irradiation. The two MT maps and two ihMT maps
are obtained using five contrasts: single frequency positive or negative and
dual frequency with an alternation of both positive and negative frequency
(saturated images); and one unsaturated contrast as reference (T1weighted).

Input Data recommendation:
  - it is recommended to use dcm2niix (v1.0.20200331) to convert data
    https://github.com/rordenlab/dcm2niix/releases/tag/v1.0.20200331
  - dcm2niix conversion will create all echo files for each contrast and
    corresponding json files
  - all input must have a matching json file with the same filename
  - all contrasts must have a same number of echoes and coregistered
    between them before running the script.
  - Mask must be coregistered to the echo images
  - ANTs can be used for the registration steps (http://stnava.github.io/ANTs/)
```

(continues on next page)

(continued from previous page)

```
The output consist in two types of images in two folders :
  1. Contrasts_ihMT_maps which contains the 5 contrast images
      - altnp.nii.gz : alternating negative and positive frequency pulses
      - altpn.nii.gz : alternating positive and negative frequency pulses
      - positive.nii.gz : pulses applied at positive frequency
      - negative.nii.gz : pulses applied at negative frequency
      - reference.nii.gz : no pulse

  2. ihMT_native_maps which contains the 4 myelin maps
      - MTR.nii.gz : Magnetization Transfer Ratio map
      - ihMTR.nii.gz : inhomogeneous Magnetization Transfer Ratio map
      The (ih)MT ratio is a measure reflecting the amount of bound protons.

      - MTsat.nii.gz : Magnetization Transfer saturation map
      - ihMTsat.nii.gz : inhomogeneous Magnetization Transfer saturation map
      The (ih)MT saturation is a pseudo-quantitative maps representing
      the signal change between the bound and free water pools.

  These final maps can be corrected by an empiric B1 correction with
  --in_B1_map option, suffix *B1_corrected is added for each map.

>>> scil_compute_ihMT_maps.py path/to/output/directory path/to/mask_bin.nii.gz
    --in_altnp path/to/echo*altnp.nii.gz --in_altpn path/to/echo*altpn.nii.gz
    --in_mtoff path/to/echo*mtoff.nii.gz --in_negative path/to/echo*neg.nii.gz
    --in_positive path/to/echo*pos.nii.gz --in_t1w path/to/echo*T1w.nii.gz

By default, the script uses all the echoes available in the input folder.
If you want to use a single echo add --single_echo to the command line and
replace the * with the specific number of the echo.

positional arguments:
  out_dir               Path to output folder.
  in_mask               Path to the T1 binary brain mask. Must be the sum of the␣
↪three tissue probability maps from T1 segmentation (GM+WM+CSF).

optional arguments:
  -h, --help            show this help message and exit
  --out_prefix OUT_PREFIX
                        Prefix to be used for each output image.
  --in_B1_map IN_B1_MAP
                        Path to B1 coregister map to MT contrasts.
  --filtering           Gaussian filtering to remove Gibbs ringing. Not recommended.
  --single_echo         Use this option when there is only one echo.
  -f                    Force overwriting of the output files.

ihMT contrasts:
  Path to echoes corresponding to contrasts images. All constrasts must have the same␣
↪number of echoes and coregistered between them. Use * to include all echoes.

  --in_altnp IN_ALTNP [IN_ALTNP ...]
                        Path to all echoes corresponding to the alternation of␣
↪Negative and Positivefrequency saturation pulse.
  --in_altpn IN_ALTPN [IN_ALTPN ...]
                        Path to all echoes corresponding to the alternation of␣
↪Positive and Negative frequency saturation pulse.
  --in_mtoff IN_MTOFF [IN_MTOFF ...]
```

(continues on next page)

(continued from previous page)

```
                        Path to all echoes corresponding to the no frequency␣
↪saturation pulse (reference image).
  --in_negative IN_NEGATIVE [IN_NEGATIVE ...]
                        Path to all echoes corresponding to the Negative frequency␣
↪saturation pulse.
  --in_positive IN_POSITIVE [IN_POSITIVE ...]
                        Path to all echoes corresponding to the Positive frequency␣
↪saturation pulse.
  --in_t1w IN_T1W [IN_T1W ...]
                        Path to all echoes corresponding to the T1-weigthed.
```

2.35 scil_compute_kurtosis_metrics.py

```
usage: __main__.py [-h] [--mask MASK] [--tolerance INT] [--min_k MIN_K]
                   [--max_k MAX_K] [--smooth SMOOTH] [--not_all] [--ak file]
                   [--mk file] [--rk file] [--msk file] [--dki_fa file]
                   [--dki_md file] [--dki_ad file] [--dki_rd file]
                   [--dki_residual file] [--msd file] [--force_b0_threshold]
                   [-f]
                   in_dwi in_bval in_bvec

Script to compute the Diffusion Kurtosis Imaging (DKI) and Mean Signal DKI
(MSDKI) metrics. DKI is a multi-shell diffusion model. The input DWI needs
to be multi-shell, i.e. multi-bvalued.

Since the diffusion kurtosis model involves the estimation of a large number
of parameters and since the non-Gaussian components of the diffusion signal
are more sensitive to artefacts, you should really denoise your DWI volume
before using this DKI script (e.g. scil_run_nlmeans.py). Moreover, to remove
biases due to fiber dispersion, fiber crossings and other mesoscopic properties
of the underlying tissue, MSDKI does a powder-average of DWI for all
directions, thus removing the orientational dependencies and creating an
alternative mean kurtosis map.

DKI is also known to be vulnerable to artefacted voxels induced by the
low radial diffusivities of aligned white matter (CC, CST voxels). Since it is
very hard to capture non-Gaussian information due to the low decays in radial
direction, its kurtosis estimates have very low robustness.
Noisy kurtosis estimates tend to be negative and its absolute values can have
order of magnitudes higher than the typical kurtosis values. Consequently,
these negative kurtosis values will heavily propagate to the mean and radial
kurtosis metrics. This is well-reported in [Rafael Henriques MSc thesis 2012,
chapter 3]. Two ways to overcome this issue: i) compute the kurtosis values
from powder-averaged MSDKI, and ii) perform 3D Gaussian smoothing. On
powder-averaged signal decays, you don't have this low diffusivity issue and
your kurtosis estimates have much higher precision (additionally they are
independent to the fODF).

By default, will output all available metrics, using default names. Specific
names can be specified using the metrics flags that are listed in the "Metrics
files flags" section. If --not_all is set, only the metrics specified
explicitly by the flags will be output.

This script directly comes from the DIPY example gallery and references
```

(continues on next page)

(continued from previous page)

```
therein.
[1] examples_built/reconst_dki/#example-reconst-dki
[2] examples_built/reconst_msdki/#example-reconst-msdki

positional arguments:
  in_dwi                Path of the input multi-shell DWI dataset.
  in_bval               Path of the b-value file, in FSL format.
  in_bvec               Path of the b-vector file, in FSL format.

optional arguments:
  -h, --help            show this help message and exit
  --mask MASK           Path to a binary mask.
                        Only data inside the mask will be used for computations and␣
↪reconstruction.
                        [Default: None]
  --tolerance INT, -t INT
                        The tolerated distance between the b-values to extract
                        and the actual b-values [Default: 20].
  --min_k MIN_K         Minimum kurtosis value in the output maps
                        (ak, mk, rk). In theory, -3/7 is the min kurtosis
                        limit for regions that consist of water confined
                        to spherical pores (see DIPY example and
                        documentation) [Default: 0.0].
  --max_k MAX_K         Maximum kurtosis value in the output maps
                        (ak, mk, rk). In theory, 10 is the max kurtosis
                        limit for regions that consist of water confined
                        to spherical pores (see DIPY example and
                        documentation) [Default: 3.0].
  --smooth SMOOTH       Smooth input DWI with a 3D Gaussian filter with
                        full-width-half-max (fwhm). Kurtosis fitting is
                        sensitive and outliers occur easily. According to
                        tests on HCP, CB_Brain, Penthera3T, this smoothing
                        is thus turned ON by default with fwhm=2.5.
                        [Default: 2.5].
  --not_all             If set, will only save the metrics explicitly
                        specified using the other metrics flags.
                        [Default: not set].
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  -f                    Force overwriting of the output files.

Metrics files flags:
  --ak file             Output filename for the axial kurtosis.
  --mk file             Output filename for the mean kurtosis.
  --rk file             Output filename for the radial kurtosis.
  --msk file            Output filename for the mean signal kurtosis.
  --dki_fa file         Output filename for the fractional anisotropy from DKI.
  --dki_md file         Output filename for the mean diffusivity from DKI.
  --dki_ad file         Output filename for the axial diffusivity from DKI.
  --dki_rd file         Output filename for the radial diffusivity from DKI.

Quality control files flags:
  --dki_residual file   Output filename for the map of the residual of the tensor fit.
  --msd file            Output filename for the mean signal diffusion (powder-
↪average).
```

2.36 scil_compute_lobe_specific_fodf_metrics.py

```
usage: __main__.py [-h] [--out_fd OUT_FD] [--out_fs OUT_FS] [--out_ff OUT_FF]
                   [--not_all] [--mask MASK]
                   [--nbr_integration_steps NBR_INTEGRATION_STEPS] [-f] [-v]
                   [--processes NBR]
                   in_bingham

Script to compute fODF lobe-specific metrics derived from a Bingham
distribution fitting, as described in [1]. Resulting metrics are fiber density
(FD), fiber spread (FS) and fiber fraction (FF) [2].

The Bingham coefficients volume comes from scil_fit_bingham_to_fodf.py.

A lobe's FD is the integral of the Bingham function on the sphere. It
represents the density of fibers going through a given voxel for a given
fODF lobe (fixel). A lobe's FS is the ratio of its FD on its maximum AFD. It
is at its minimum for a sharp lobe and at its maximum for a wide lobe. A lobe's
FF is the ratio of its FD on the total FD in the voxel.

Using 12 threads, the execution takes 10 minutes for FD estimation for a brain
with 1mm isotropic resolution. Other metrics take less than a second.

positional arguments:
  in_bingham            Input Bingham image.

optional arguments:
  -h, --help            show this help message and exit
  --out_fd OUT_FD       Path to output fiber density. [fd.nii.gz]
  --out_fs OUT_FS       Path to output fiber spread. [fs.nii.gz]
  --out_ff OUT_FF       Path to fiber fraction file. [ff.nii.gz]
  --not_all             Do not compute all metrics.
  --mask MASK           Optional mask image. Only voxels inside the mask are computed.
  --nbr_integration_steps NBR_INTEGRATION_STEPS
                        Number of integration steps along the theta axis for fiber␣
↪density estimation. [50]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]

[1] T. W. Riffert, J. Schreiber, A. Anwander, and T. R. Knösche, "Beyond
    fractional anisotropy: Extraction of bundle-specific structural metrics
    from crossing fiber models," NeuroImage, vol. 100, pp. 176-191, Oct. 2014,
    doi: 10.1016/j.neuroimage.2014.06.015.

[2] J. Schreiber, T. Riffert, A. Anwander, and T. R. Knösche, "Plausibility
    Tracking: A method to evaluate anatomical connectivity and microstructural
    properties along fiber pathways," NeuroImage, vol. 90, pp. 163-178, Apr.
    2014, doi: 10.1016/j.neuroimage.2014.01.002.
```

2.37 scil_compute_local_tracking.py

```
usage: __main__.py [-h] [--step STEP_SIZE] [--min_length m] [--max_length M]
                   [--theta THETA] [--sfthres sf_th]
                   [--sh_basis {descoteaux07,tournier07}]
                   [--algo {det,prob,eudx}]
                   [--sphere {symmetric362,symmetric642,symmetric724}]
                   [--npv NPV | --nt NT] [--compress thresh] [-f]
                   [--save_seeds] [--seed SEED] [-v]
                   in_odf in_seed in_mask out_tractogram

Local streamline HARDI tractography.
The tracking direction is chosen in the aperture cone defined by the
previous tracking direction and the angular constraint.

Algo 'eudx': the peak from the spherical function (SF) most closely aligned
to the previous direction.
Algo 'det': the maxima of the spherical function (SF) the most closely aligned
to the previous direction.
Algo 'prob': a direction drawn from the empirical distribution function defined
from the SF.

NOTE: eudx can be used with pre-computed peaks from fodf as well as
evecs_v1.nii.gz from scil_compute_dti_metrics.py (experimental).

All the input nifti files must be in isotropic resolution.

positional arguments:
  in_odf                File containing the orientation diffusion function
                        as spherical harmonics file (.nii.gz). Ex: ODF or fODF.
  in_seed               Seeding mask (.nii.gz).
  in_mask               Tracking mask (.nii.gz).
                        Tracking will stop outside this mask.
  out_tractogram        Tractogram output file (must be .trk or .tck).

optional arguments:
  -h, --help            show this help message and exit

Tracking options:
  --step STEP_SIZE      Step size in mm. [0.5]
  --min_length m        Minimum length of a streamline in mm. [10.0]
  --max_length M        Maximum length of a streamline in mm. [300.0]
  --theta THETA         Maximum angle between 2 steps.
                        ["eudx"=60, "det"=45, "prob"=20]
  --sfthres sf_th       Spherical function relative threshold. [0.1]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --algo {det,prob,eudx}
                        Algorithm to use. [prob]
  --sphere {symmetric362,symmetric642,symmetric724}
                        Dipy sphere; set of possible directions.
                        Default: [symmetric724]
```

(continues on next page)

(continued from previous page)

```
Seeding options:
  When no option is provided, uses --npv 1.

  --npv NPV             Number of seeds per voxel.
  --nt NT               Total number of seeds to use.

Output options:
  --compress thresh     If set, will compress streamlines. The parameter value is the
                        distance threshold. A rule of thumb is to set it to 0.1mm for
                        deterministic streamlines and 0.2mm for probabilitic␣
↪streamlines.
  -f                    Force overwriting of the output files.
  --save_seeds          If set, save the seeds used for the tracking
                         in the data_per_streamline property.
                        Hint: you can then use scilpy_compute_seed_density_map.
  --seed SEED           Random number generator seed.

Logging options:
  -v                    If set, produces verbose output.
```

2.38 scil_compute_local_tracking_dev.py

```
usage: __main__.py [-h] [--step STEP_SIZE] [--min_length m] [--max_length M]
                   [--theta THETA] [--sfthres sf_th]
                   [--sh_basis {descoteaux07,tournier07}] [--algo {det,prob}]
                   [--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724}]
                   [--sfthres_init sf_th] [--rk_order K]
                   [--max_invalid_length MAX] [--forward_only]
                   [--sh_interp {nearest,trilinear}]
                   [--mask_interp {nearest,trilinear}] [--npv NPV | --nt NT]
                   [--rng_seed RNG_SEED] [--skip SKIP] [--processes NBR]
                   [--compress thresh] [-f] [--save_seeds] [-v]
                   in_odf in_seed in_mask out_tractogram

Local streamline HARDI tractography using scilpy-only methods -- no dipy (i.e
no cython). The goal of this is to have a python-only version that can be
modified more easily by our team when testing new algorithms and parameters,
and that can be used as parent classes in sub-projects of our lab such as in
dwi_ml.

As in scil_compute_local_tracking:

    The tracking direction is chosen in the aperture cone defined by the
    previous tracking direction and the angular constraint.
    - Algo 'det': the maxima of the spherical function (SF) the most closely
    aligned to the previous direction.
    - Algo 'prob': a direction drawn from the empirical distribution function
    defined from the SF.
    - Algo 'eudx' is not yet available!

Contrary to scil_compute_local_tracking:
    - Input nifti files do not necessarily need to be in isotropic resolution.
```

(continues on next page)

(continued from previous page)

```
    - The script works with asymmetric input ODF.
    - The interpolation for the tracking mask and spherical function can be
      one of 'nearest' or 'trilinear'.
    - Runge-Kutta integration is supported for the step function.

A few notes on Runge-Kutta integration.
    1. Runge-Kutta integration is used to approximate the next tracking
       direction by estimating directions from future tracking steps. This
       works well for deterministic tracking. However, in the context of
       probabilistic tracking, the next tracking directions cannot be estimated
       in advance, because they are picked randomly from a distribution. It is
       therefore recommanded to keep the rk_order to 1 for probabilistic
       tracking.
    2. As a rule of thumb, doubling the rk_order will double the computation
       time in the worst case.

References: [1] Girard, G., Whittingstall K., Deriche, R., and
            Descoteaux, M. (2014). Towards quantitative connectivity analysis:
            reducing tractography biases. Neuroimage, 98, 266-278.

positional arguments:
  in_odf                File containing the orientation diffusion function
                        as spherical harmonics file (.nii.gz). Ex: ODF or fODF.
  in_seed               Seeding mask (.nii.gz).
  in_mask               Tracking mask (.nii.gz).
                        Tracking will stop outside this mask.
  out_tractogram        Tractogram output file (must be .trk or .tck).

optional arguments:
  -h, --help            show this help message and exit
  -v                    If set, produces verbose output.

Tracking options:
  --step STEP_SIZE      Step size in mm. [0.5]
  --min_length m        Minimum length of a streamline in mm. [10.0]
  --max_length M        Maximum length of a streamline in mm. [300.0]
  --theta THETA         Maximum angle between 2 steps.
                        ["eudx"=60, "det"=45, "prob"=20]
  --sfthres sf_th       Spherical function relative threshold. [0.1]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --algo {det,prob}     Algorithm to use. [prob]
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Dipy sphere; set of possible directions.
                        Default: [symmetric724]
  --sfthres_init sf_th  Spherical function relative threshold value for the
                        initial direction. [0.5]
  --rk_order K          The order of the Runge-Kutta integration used for the step␣
↪function.
                        For more information, refer to the note in the script␣
↪description. [1]
```

(continues on next page)

(continued from previous page)

```
  --max_invalid_length MAX
                        Maximum length without valid direction, in mm. [1]
  --forward_only        If set, tracks in one direction only (forward) given the
                        initial seed. The direction is randomly drawn from the ODF.
  --sh_interp {nearest,trilinear}
                        Spherical harmonic interpolation: nearest-neighbor
                        or trilinear. [trilinear]
  --mask_interp {nearest,trilinear}
                        Mask interpolation: nearest-neighbor or trilinear. [nearest]

Seeding options:
  When no option is provided, uses --npv 1.

  --npv NPV             Number of seeds per voxel.
  --nt NT               Total number of seeds to use.

Random seeding options:
  --rng_seed RNG_SEED   Initial value for the random number generator. [0]
  --skip SKIP           Skip the first N random number.
                        Useful if you want to create new streamlines to add to
                        a previously created tractogram with a fixed --rng_seed.
                        Ex: If tractogram_1 was created with -nt 1,000,000,
                        you can create tractogram_2 with
                        --skip 1,000,000.

Memory options:
  --processes NBR       Number of sub-processes to start.
                        Default: [1]

Output options:
  --compress thresh     If set, will compress streamlines. The parameter value is the
                        distance threshold. A rule of thumb is to set it to 0.1mm for
                        deterministic streamlines and 0.2mm for probabilitic␣
↪streamlines.
  -f                    Force overwriting of the output files.
  --save_seeds          If set, save the seeds used for the tracking
                         in the data_per_streamline property.
                        Hint: you can then use scilpy_compute_seed_density_map.
```

2.39 scil_compute_local_tracking_gpu.py

```
usage: __main__.py [-h] [--step_size STEP_SIZE] [--theta THETA [THETA ...]]
                   [--min_length MIN_LENGTH] [--max_length MAX_LENGTH]
                   [--sf_threshold SF_THRESHOLD]
                   [--sh_interp {nearest,trilinear}]
                   [--mask_interp {nearest,trilinear}] [--forward_only]
                   [--sh_basis {descoteaux07,tournier07}]
                   [--npv NPV | --nt NT] [--compress thresh] [-f]
                   [--save_seeds] [--rng_seed RNG_SEED]
                   [--batch_size BATCH_SIZE] [-v]
                   in_odf in_seed in_mask out_tractogram

Perform probabilistic tractography on a ODF field inside a binary mask.
The tracking is executed on the GPU using the OpenCL API.
```

(continues on next page)

(continued from previous page)

```
Streamlines are filtered by minimum length, but not by maximum length. This
means that streamlines are stopped and returned as soon as they reach the
maximum length instead of being discarded if they go above the maximum allowed
length. This allows for short-tracks tractography, where streamlines are
prematurely stopped once they reach some target length.

However, for this reason, there may be streamlines ending in the deep white
matter. In order to use the resulting tractogram for analysis, it should be
cleaned with scil_filter_tractogram_anatomically.py.

The white matter mask is interpolated using nearest-neighbor interpolation and
the SH interpolation defaults to trilinear.

The script also incorporates ideas from Ensemble Tractography [1] (ET). Given
a list of maximum angles, a different angle drawn at random from the set will
be used for each streamline.

In order to use the script, you must have a OpenCL compatible GPU and install
the pyopencl package via `pip install pyopencl`.

positional arguments:
  in_odf                File containing the orientation diffusion function
                        as spherical harmonics file (.nii.gz). Ex: ODF or fODF.
  in_seed               Seeding mask (.nii.gz).
  in_mask               Tracking mask (.nii.gz).
                        Tracking will stop outside this mask.
  out_tractogram        Tractogram output file (must be .trk or .tck).

Generic options:
  -h, --help            show this help message and exit

Tracking options:
  --step_size STEP_SIZE
                        Step size in mm. [0.5]
  --theta THETA [THETA ...]
                        Maximum angle between 2 steps. If more than one value
                        are given, the maximum angle will be drawn at random
                        from the distribution for each streamline. [20.0]
  --min_length MIN_LENGTH
                        Minimum length of the streamline in mm. [20.0]
  --max_length MAX_LENGTH
                        Maximum length of the streamline in mm. [300.0]
  --sf_threshold SF_THRESHOLD
                        Relative threshold on sf amplitudes. [0.1]
  --sh_interp {nearest,trilinear}
                        SH interpolation mode. [trilinear]
  --mask_interp {nearest,trilinear}
                        Mask interpolation. Only nearest-neighbour interpolation
                        is available for now. [nearest]
  --forward_only        Only perform forward tracking.
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
```

(continues on next page)

(continued from previous page)

```
                                                NeuroImage 2007 paper.

Seeding options:
  When no option is provided, uses --npv 1.

  --npv NPV             Number of seeds per voxel.
  --nt NT               Total number of seeds to use.

Output options:
  --compress thresh     If set, will compress streamlines. The parameter value is the
                        distance threshold. A rule of thumb is to set it to 0.1mm for
                        deterministic streamlines and 0.2mm for probabilitic␣
↪streamlines.
  -f                    Force overwriting of the output files.
  --save_seeds          If set, save the seeds used for the tracking
                         in the data_per_streamline property.
                        Hint: you can then use scilpy_compute_seed_density_map.
  --rng_seed RNG_SEED   Random number generator seed.

GPU options:
  --batch_size BATCH_SIZE
                        Approximate size of GPU batches (number
                        of streamlines to track in parallel). [100000]

Logging options:
  -v                    If set, produces verbose output.

[1] Takemura, H. et al (2016). Ensemble tractography. PLoS Computational
    Biology, 12(2), e1004692.
```

2.40 scil_compute_maps_for_particle_filter_tracking.py

```
usage: __main__.py [-h] [--include filename] [--exclude filename]
                   [--interface filename] [-t THRESHOLD] [-f] [-v]
                   in_wm in_gm in_csf

Compute include and exclude maps, and the seeding interface mask from partial
volume estimation (PVE) maps. Maps should have values in [0,1], gm+wm+csf=1 in
all voxels of the brain, gm+wm+csf=0 elsewhere.

References: Girard, G., Whittingstall K., Deriche, R., and Descoteaux, M.
(2014). Towards quantitative connectivity analysis: reducing tractography
biases. Neuroimage.

positional arguments:
  in_wm                 White matter PVE map (nifti). From normal FAST output, has a␣
↪PVE_2 name suffix.
  in_gm                 Grey matter PVE map (nifti). From normal FAST output, has a␣
↪PVE_1 name suffix.
  in_csf                Cerebrospinal fluid PVE map (nifti). From normal FAST output,␣
↪has a PVE_0 name suffix.

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --include filename    Output include map (nifti). [map_include.nii.gz]
  --exclude filename    Output exclude map (nifti). [map_exclude.nii.gz]
  --interface filename  Output interface seeding mask (nifti). [interface.nii.gz]
  -t THRESHOLD          Minimum gm and wm PVE values in a voxel to be into the␣
↪interface. [0.1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.41 scil_compute_mean_fixel_afd_from_bundles.py

```
usage: __main__.py [-h] [--length_weighting] [--reference REFERENCE]
                   [--sh_basis {descoteaux07,tournier07}] [-f]
                   in_bundle in_fodf afd_mean_map

Compute the mean Apparent Fiber Density (AFD) and mean Radial fODF (radfODF)
maps along a bundle.

This is the "real" fixel-based fODF amplitude along every streamline
of the bundle provided, averaged at every voxel.

Please use a bundle file rather than a whole tractogram.

positional arguments:
  in_bundle             Path of the bundle file.
  in_fodf               Path of the fODF volume in spherical harmonics (SH).
  afd_mean_map          Path of the output mean AFD map.

optional arguments:
  -h, --help            show this help message and exit
  --length_weighting    If set, will weigh the AFD values according to segment␣
↪lengths. [False]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -f                    Force overwriting of the output files.

Reference:
    [1] Raffelt, D., Tournier, JD., Rose, S., Ridgway, GR., Henderson, R.,
        Crozier, S., Salvado, O., & Connelly, A. (2012).
        Apparent Fibre Density: a novel measure for the analysis of
        diffusion-weighted magnetic resonance images. NeuroImage, 59(4), 3976--3994.
```

2.42 scil_compute_mean_fixel_afd_from_hdf5.py

```
usage: __main__.py [-h] [--length_weighting] [--processes NBR]
                   [--sh_basis {descoteaux07,tournier07}] [-f]
                   in_hdf5 in_fodf out_hdf5

Compute the mean Apparent Fiber Density (AFD) and mean Radial fODF (radfODF)
maps along a bundle.

This is the "real" fixel-based fODF amplitude along every streamline
of the bundle provided, averaged at every voxel.

Please use a bundle file rather than a whole tractogram.

positional arguments:
  in_hdf5               HDF5 filename (.h5) containing decomposed connections.
  in_fodf               Path of the fODF volume in spherical harmonics (SH).
  out_hdf5              Path of the output HDF5 filenames (.h5).

optional arguments:
  -h, --help            show this help message and exit
  --length_weighting    If set, will weigh the AFD values according to segment␣
↪lengths. [False]
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -f                    Force overwriting of the output files.

Reference:
    [1] Raffelt, D., Tournier, JD., Rose, S., Ridgway, GR., Henderson, R.,
        Crozier, S., Salvado, O., & Connelly, A. (2012).
        Apparent Fibre Density: a novel measure for the analysis of
        diffusion-weighted magnetic resonance images. NeuroImage,
        59(4), 3976--3994.
```

2.43 scil_compute_mean_fixel_lobe_metric_from_bundles.py

```
usage: __main__.py [-h] [--length_weighting] [--max_theta MAX_THETA]
                   [--reference REFERENCE] [-f]
                   in_bundle in_bingham in_lobe_metric out_mean_map

Given a bundle and Bingham coefficients, compute the average lobe-specific
metric at each voxel intersected by the bundle. Intersected voxels are
found by computing the intersection between the voxel grid and each streamline
in the input tractogram.

This script behaves like scil_compute_mean_fixel_afd_from_bundles.py for fODFs,
but here for Bingham distributions. These latest distributions add the unique
```

(continues on next page)

(continued from previous page)

```
possibility to capture fixel-based fiber spread (FS) and fiber fraction (FF).
FD from the bingham should be "equivalent" to the AFD_fixel we are used to.

Bingham coefficients volume must come from scil_fit_bingham_to_fodf.py and
lobe-specific metrics comes from scil_compute_lobe_specific_fodf_metrics.py.

Lobe-specific metrics are metrics extracted from Bingham distributions fitted
to fODF. Their are as many values per voxel as there are lobes extracted. The
values chosen for a given voxelis the one belonging to the lobe better aligned
with the current streamline segment.

Please use a bundle file rather than a whole tractogram.

positional arguments:
  in_bundle             Path of the bundle file.
  in_bingham            Path of the Bingham volume.
  in_lobe_metric        Path of the lobe-specific metric (FD, FS, or FF) volume.
  out_mean_map          Path of the output mean map.

optional arguments:
  -h, --help            show this help message and exit
  --length_weighting    If set, will weigh the FD values according to segment lengths.
↪ [False]
  --max_theta MAX_THETA
                        Maximum angle (in degrees) condition on lobe alignment. [60]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.44 scil_compute_mean_frf.py

```
usage: __main__.py [-h] [-f] list [list ...] file

Compute the mean Fiber Response Function from a set of individually
computed Response Functions.

positional arguments:
  list        List of FRF filepaths.
  file        Path of the output mean FRF file.

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
```

2.45 scil_compute_memsmt_fodf.py

```
usage: __main__.py [-h] --in_dwis IN_DWIS [IN_DWIS ...] --in_bvals IN_BVALS
                   [IN_BVALS ...] --in_bvecs IN_BVECS [IN_BVECS ...]
                   --in_bdeltas {0,1,-0.5,0.5} [{0,1,-0.5,0.5} ...]
                   [--sh_order int] [--mask MASK] [--tolerance TOLERANCE]
```

(continues on next page)

(continued from previous page)

```
                    [--force_b0_threshold]
                    [--sh_basis {descoteaux07,tournier07}] [--processes NBR]
                    [-f] [-v] [--not_all] [--wm_out_fODF file]
                    [--gm_out_fODF file] [--csf_out_fODF file] [--vf file]
                    [--vf_rgb file]
                    in_wm_frf in_gm_frf in_csf_frf

Script to compute multi-encoding multi-shell multi-tissue (memsmt)
Constrained Spherical Deconvolution ODFs. In order to operate,
the script only needs the data from one type of b-tensor encoding. However,
giving only a spherical one will not produce good fODFs, as it only probes
spherical shapes. As for planar encoding, it should technically
work alone, but seems to be very sensitive to noise and is yet to be properly
documented. We thus suggest to always use at least the linear encoding, which
will be equivalent to standard multi-shell multi-tissue if used alone, in
combinaison with other encodings. Note that custom encodings are not yet
supported, so that only the linear tensor encoding (LTE, b_delta = 1), the
planar tensor encoding (PTE, b_delta = -0.5), the spherical tensor encoding
(STE, b_delta = 0) and the cigar shape tensor encoding (b_delta = 0.5) are
available. Moreover, all of `--in_dwis`, `--in_bvals`, `--in_bvecs` and
`--in_bdeltas` must have the same number of arguments. Be sure to keep the
same order of encodings throughout all these inputs and to set `--in_bdeltas`
accordingly (IMPORTANT).

By default, will output all possible files, using default names.
Specific names can be specified using the file flags specified in the
"File flags" section.

If --not_all is set, only the files specified explicitly by the flags
will be output.

Based on P. Karan et al., Bridging the gap between constrained spherical
deconvolution and diffusional variance decomposition via tensor-valued
diffusion MRI. Medical Image Analysis (2022)

positional arguments:
  in_wm_frf             Text file of WM response function.
  in_gm_frf             Text file of GM response function.
  in_csf_frf            Text file of CSF response function.

optional arguments:
  -h, --help            show this help message and exit
  --in_dwis IN_DWIS [IN_DWIS ...]
                        Path to the input diffusion volume for each b-tensor encoding␣
↪type.
  --in_bvals IN_BVALS [IN_BVALS ...]
                        Path to the bval file, in FSL format, for each b-tensor␣
↪encoding type.
  --in_bvecs IN_BVECS [IN_BVECS ...]
                        Path to the bvec file, in FSL format, for each b-tensor␣
↪encoding type.
  --in_bdeltas {0,1,-0.5,0.5} [{0,1,-0.5,0.5} ...]
                        Value of b_delta for each b-tensor encoding type, in the same␣
↪order as dwi, bval and bvec inputs.
  --sh_order int        SH order used for the CSD. (Default: 8)
  --mask MASK           Path to a binary mask. Only the data inside the mask will be␣
↪used for computations and reconstruction.
```

(continues on next page)

(continued from previous page)

```
  --tolerance TOLERANCE
                        The tolerated gap between the b-values to extract
                        and the current b-value. [20]
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --not_all             If set, only saves the files specified using the file flags.␣
↪(Default: False)

File flags:
  --wm_out_fODF file    Output filename for the WM fODF coefficients.
  --gm_out_fODF file    Output filename for the GM fODF coefficients.
  --csf_out_fODF file   Output filename for the CSF fODF coefficients.
  --vf file             Output filename for the volume fractions map.
  --vf_rgb file         Output filename for the volume fractions map in rgb.
```

2.46 scil_compute_memsmt_frf.py

2.47 scil_compute_metrics_stats_in_ROI.py

```
usage: __main__.py [-h]
                   (--metrics_dir METRICS_DIR | --metrics METRICS_FILE_LIST [METRICS_
↪FILE_LIST ...])
                   [--bin] [--normalize_weights] [-f] [--indent INDENT]
                   [--sort_keys]
                   in_mask

Compute the statistics (mean, std) of scalar maps, which can represent
diffusion metrics, in a ROI.

The mask can either be a binary mask, or a weighting mask. If the mask is
a weighting mask it should either contain floats between 0 and 1 or should be
normalized with --normalize_weights.

IMPORTANT: if the mask contains weights (and not 0 and 1 exclusively), the
standard deviation will also be weighted.

positional arguments:
  in_mask               Mask volume filename.
                        Can be a binary mask or a weighted mask.

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --bin                 If set, will consider every value of the mask higher than 0␣
↪to be part of the mask, and set to 1 (equivalent weighting for every voxel).
  --normalize_weights   If set, the weights will be normalized to the [0,1] range.
  -f                    Force overwriting of the output files.

Metrics input options:
  --metrics_dir METRICS_DIR
                        Metrics files directory. Name of the directory containing the␣
↪metrics files.
  --metrics METRICS_FILE_LIST [METRICS_FILE_LIST ...]
                        Metrics nifti filename. List of the names of the metrics file,
↪ in nifti format.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.48 scil_compute_msmt_fodf.py

```
usage: __main__.py [-h] [--sh_order int] [--mask] [--force_b0_threshold]
                   [--sh_basis {descoteaux07,tournier07}] [--processes NBR]
                   [-f] [-v] [--not_all] [--wm_out_fODF file]
                   [--gm_out_fODF file] [--csf_out_fODF file] [--vf file]
                   [--vf_rgb file]
                   in_dwi in_bval in_bvec in_wm_frf in_gm_frf in_csf_frf

Script to compute Multishell Multi-tissue Constrained Spherical Deconvolution
ODFs.

By default, will output all possible files, using default names.
Specific names can be specified using the file flags specified in the
"File flags" section.

If --not_all is set, only the files specified explicitly by the flags
will be output.

Based on B. Jeurissen et al., Multi-tissue constrained spherical
deconvolution for improved analysis of multi-shell diffusion
MRI data. Neuroimage (2014)

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bval file, in FSL format.
  in_bvec               Path of the bvec file, in FSL format.
  in_wm_frf             Text file of WM response function.
  in_gm_frf             Text file of GM response function.
  in_csf_frf            Text file of CSF response function.

optional arguments:
  -h, --help            show this help message and exit
  --sh_order int        SH order used for the CSD. (Default: 8)
  --mask                Path to a binary mask. Only the data inside the mask will be␣
↪used for computations and reconstruction.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
```

(continues on next page)

(continued from previous page)

```
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --not_all             If set, only saves the files specified using the file flags.␣
↪(Default: False)

File flags:
  --wm_out_fODF file    Output filename for the WM fODF coefficients.
  --gm_out_fODF file    Output filename for the GM fODF coefficients.
  --csf_out_fODF file   Output filename for the CSF fODF coefficients.
  --vf file             Output filename for the volume fractions map.
  --vf_rgb file         Output filename for the volume fractions map in rgb.
```

2.49 scil_compute_msmt_frf.py

2.50 scil_compute_pca.py

```
usage: __main__.py [-h] --metrics METRICS [METRICS ...] --list_ids FILE
                   [--not_only_common] [--input_connectoflow] [-v] [-f]
                   in_folder out_folder

Script to compute PCA analysis on diffusion metrics. Output returned is all␣
↪significant principal components
(e.g. presenting eigenvalues > 1) in a connectivity matrix format. This script can␣
↪take into account all
edges from every subject in a population or only non-zero edges across all subjects.

The script can take directly as input a connectoflow output folder. Simply use the --
↪input_connectoflow flag.
For other type of folder input, the script expects a single folder containing all␣
↪matrices for all subjects.
Example:
        [in_folder]
        |--- sub-01_ad.npy
        |--- sub-01_md.npy
        |--- sub-02_ad.npy
        |--- sub-02_md.npy
        |--- ...

The plots, tables and principal components matrices will be outputted in the␣
↪designated folder from the
<out_folder> argument. If you want to move back your principal components matrices in␣
↪your connectoflow
output, you can use a similar bash command for all principal components:
for sub in `cat list_id.txt`; do cp out_folder/${sub}_PC1.npy connectoflow_output/
↪$sub/Compute_Connectivity/; done
```

(continues on next page)

(continued from previous page)

```
Interpretation of resulting principal components can be done by evaluating the␣
↪loadings values for each metrics.
A value near 0 means that this metric doesn't contribute to this specific component␣
↪whereas high positive or
negative values mean a larger contribution. Components can then be labeled based on␣
↪which metric contributes
the highest. For example, a principal component showing a high loading for afd_fixel␣
↪and near 0 loading for all
other metrics can be interpreted as axonal density (see Gagnon et al. 2022 for this␣
↪specific example or
ref [3] for an introduction to PCA).

EXAMPLE USAGE:
scil_compute_pca.py input_folder/ output_folder/ --metrics ad fa md rd [...] --list_
↪ids list_ids.txt

positional arguments:
  in_folder             Path to the input folder.
  out_folder            Path to the output folder to export graphs, tables and␣
↪principal components matrices.

optional arguments:
  -h, --help            show this help message and exit
  --metrics METRICS [METRICS ...]
                        Suffixes of all metrics to include in PCA analysis (ex: ad md␣
↪fa rd). They must be immediately followed by the .npy extension.
  --list_ids FILE       Path to a .txt file containing a list of all ids.
  --not_only_common     If true, will include all edges from all subjects and not␣
↪only common edges (Not recommended)
  --input_connectoflow  If true, script will assume the input folder is a␣
↪Connectoflow output.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

[1] Chamberland M, Raven EP, Genc S, Duffy K, Descoteaux M, Parker GD, Tax CMW, Jones␣
↪DK. Dimensionality
    reduction of diffusion MRI measures for improved tractometry of the human brain.␣
↪Neuroimage. 2019 Oct
    15;200:89-100. doi: 10.1016/j.neuroimage.2019.06.020. Epub 2019 Jun 20. PMID:␣
↪31228638; PMCID: PMC6711466.
[2] Gagnon A., Grenier G., Bocti C., Gillet V., Lepage J.-F., Baccarelli A. A.,␣
↪Posner J., Descoteaux M.,
    & Takser L. (2022). White matter microstructural variability linked to␣
↪differential attentional skills
    and impulsive behavior in a pediatric population. Cerebral Cortex. https://doi.
↪org/10.1093/cercor/bhac180
[3] https://towardsdatascience.com/what-are-pca-loadings-and-biplots-9a7897f2e559
```

2.51 scil_compute_pft.py

```
usage: __main__.py [-h] [--algo {det,prob}] [--step STEP_SIZE]
                   [--min_length MIN_LENGTH] [--max_length MAX_LENGTH]
                   [--theta THETA] [--act] [--sfthres SF_THRESHOLD]
```

(continues on next page)

(continued from previous page)

```
                    [--sfthres_init SF_THRESHOLD_INIT]
                    [--sh_basis {descoteaux07,tournier07}]
                    [--npv NPV | --nt NT] [--particles PARTICLES]
                    [--back BACK_TRACKING] [--forward FORWARD_TRACKING]
                    [--compress COMPRESS] [--all] [--seed SEED] [-f]
                    [--save_seeds] [-v]
                    in_sh in_seed in_map_include map_exclude_file
                    out_tractogram

Local streamline HARDI tractography including Particle Filtering tracking.

The tracking is done inside partial volume estimation maps and uses the
particle filtering tractography (PFT) algorithm. See
scil_compute_maps_for_particle_filter_tracking.py
to generate PFT required maps.

Streamlines longer than min_length and shorter than max_length are kept.
The tracking direction is chosen in the aperture cone defined by the
previous tracking direction and the angular constraint.
Default parameters as suggested in [1].

Algo 'det': the maxima of the spherical function (SF) the most closely aligned
to the previous direction.
Algo 'prob': a direction drawn from the empirical distribution function defined
from the SF.

For streamline compression, a rule of thumb is to set it to 0.1mm for the
deterministic algorithm and 0.2mm for probabilitic algorithm.

All the input nifti files must be in isotropic resolution.

positional arguments:
  in_sh                 Spherical harmonic file (.nii.gz).
  in_seed               Seeding mask (.nii.gz).
  in_map_include        The probability map (.nii.gz) of ending the streamline
                        and including it in the output (CMC, PFT [1])
  map_exclude_file      The probability map (.nii.gz) of ending the streamline
                        and excluding it in the output (CMC, PFT [1]).
  out_tractogram        Tractogram output file (must be .trk or .tck).

Generic options:
  -h, --help            show this help message and exit

Tracking options:
  --algo {det,prob}     Algorithm to use (must be "det" or "prob"). [prob]
  --step STEP_SIZE      Step size in mm. [0.2]
  --min_length MIN_LENGTH
                        Minimum length of a streamline in mm. [10.0]
  --max_length MAX_LENGTH
                        Maximum length of a streamline in mm. [300.0]
  --theta THETA         Maximum angle between 2 steps. ["det"=45, "prob"=20]
  --act                 If set, uses anatomically-constrained tractography (ACT)
                        instead of continuous map criterion (CMC).
  --sfthres SF_THRESHOLD
                        Spherical function relative threshold. [0.1]
  --sfthres_init SF_THRESHOLD_INIT
                        Spherical function relative threshold value for the
```

(continues on next page)

(continued from previous page)

```
                        initial direction. [0.5]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.

Seeding options:
  When no option is provided, uses --npv 1.

  --npv NPV             Number of seeds per voxel.
  --nt NT               Total number of seeds to use.

PFT options:
  --particles PARTICLES
                        Number of particles to use for PFT. [15]
  --back BACK_TRACKING  Length of PFT back tracking (mm). [2.0]
  --forward FORWARD_TRACKING
                        Length of PFT forward tracking (mm). [1.0]

Output options:
  --compress COMPRESS   If set, will compress streamlines.
                        The parameter value is the distance threshold.
  --all                 If set, keeps "excluded" streamlines.
                        NOT RECOMMENDED, except for debugging.
  --seed SEED           Random number generator seed.
  -f                    Force overwriting of the output files.
  --save_seeds          If set, save the seeds used for the tracking
                         in the data_per_streamline property.

Logging options:
  -v                    If set, produces verbose output.

References: [1] Girard, G., Whittingstall K., Deriche, R., and Descoteaux, M. (2014).␣
↪Towards quantitative connectivity analysis: reducing tractography biases.␣
↪Neuroimage, 98, 266-278.
```

2.52 scil_compute_powder_average.py

```
usage: __main__.py [-h] [-f] [--mask file] [--b0_thr B0_THR]
                   [--shells SHELLS [SHELLS ...]] [--shell_thr SHELL_THR] [-v]
                   in_dwi in_bval out_avg

Script to compute powder average (mean diffusion weighted image) from set of
diffusion images.

By default will output an average image calculated from all images with
non-zero bvalue.

specify --bvalue to output an image for a single shell

Script currently does not take into account the diffusion gradient directions
```

(continues on next page)

(continued from previous page)

```
being averaged.

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  out_avg               Path of the output file.

optional arguments:
  -h, --help            show this help message and exit
  -f                    Force overwriting of the output files.
  --mask file           Path to a binary mask.
                        Only data inside the mask will be used for powder avg.␣
↪(Default: None)
  --b0_thr B0_THR       Exclude b0 volumes from powder average with bvalue less than␣
↪specified threshold.
                        (Default: remove volumes with bvalue < 50
  --shells SHELLS [SHELLS ...]
                        bvalue (shells) to include in powder average passed as a list
                        (e.g. --shells 1000 2000). If not specified will include all␣
↪volumes with a non-zero bvalue.
  --shell_thr SHELL_THR
                        Include volumes with bvalue +- the specified threshold.
                        (Default: [50]
  -v                    If set, produces verbose output.
```

2.53 scil_compute_qball_metrics.py

```
usage: __main__.py [-h] [-f] [--sh_order SH_ORDER] [--mask MASK] [--use_qball]
                   [--not_all] [--gfa GFA] [--peaks PEAKS]
                   [--peak_indices PEAK_INDICES] [--sh SH] [--nufo NUFO]
                   [--a_power A_POWER] [--force_b0_threshold]
                   [--sh_basis {descoteaux07,tournier07}] [--processes NBR]
                   in_dwi in_bval in_bvec

Script to compute the Constant Solid Angle (CSA) or Analytical Q-ball model,
the generalized fractional anisotropy (GFA) and the peaks of the model.

By default, will output all possible files, using default names. Specific names
can be specified using the file flags specified in the "File flags" section.

If --not_all is set, only the files specified explicitly by the flags
will be output.

See [Descoteaux et al MRM 2007, Aganj et al MRM 2009] for details and
[Cote et al MEDIA 2013] for quantitative comparisons.

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  in_bvec               Path of the bvecs file, in FSL format.

optional arguments:
  -h, --help            show this help message and exit
  -f                    Force overwriting of the output files.
```

(continues on next page)

(continued from previous page)

```
  --sh_order SH_ORDER   Spherical harmonics order. Must be a positive even number [4].
  --mask MASK           Path to a binary mask. Only data inside the mask will be used␣
↪for computations and reconstruction [None].
  --use_qball           If set, qball will be used as the odf reconstruction model␣
↪instead of CSA.
  --not_all             If set, will only save the files specified using the␣
↪following flags.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]

File flags:
  --gfa GFA             Output filename for the generalized fractional anisotropy␣
↪[gfa.nii.gz].
  --peaks PEAKS         Output filename for the extracted peaks [peaks.nii.gz].
  --peak_indices PEAK_INDICES
                        Output filename for the generated peaks indices on the sphere␣
↪[peaks_indices.nii.gz].
  --sh SH               Output filename for the spherical harmonics coefficients [sh.
↪nii.gz].
  --nufo NUFO           Output filename for the NUFO map [nufo.nii.gz].
  --a_power A_POWER     Output filename for the anisotropic power map[anisotropic_
↪power.nii.gz].
```

2.54 scil_compute_qbx.py

```
usage: __main__.py [-h] [--nb_points NB_POINTS]
                   [--out_centroids OUT_CENTROIDS] [--reference REFERENCE]
                   [-f] [-v]
                   in_tractogram dist_thresh out_clusters_dir

    Compute clusters using QuickBundlesX and save them separately.
    We cannot know the number of clusters in advance.

positional arguments:
  in_tractogram         Tractogram filename.
                        Path of the input tractogram or bundle.
  dist_thresh           Last QuickBundlesX threshold in mm. Typically
                        the value are between 10-20mm.
  out_clusters_dir      Path to the clusters directory.

optional arguments:
  -h, --help            show this help message and exit
  --nb_points NB_POINTS
                        Streamlines will be resampled to have this number of points␣
↪[20].
```

(continues on next page)

(continued from previous page)

```
  --out_centroids OUT_CENTROIDS
                        Output tractogram filename.
                        Format must be readable by the Nibabel API.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.55 scil_compute_rish_from_sh.py

```
usage: __main__.py [-h] [--full_basis] [--mask MASK] [-f] in_sh out_prefix

Compute the RISH (Rotationally Invariant Spherical Harmonics) features
of an SH signal [1].

Each RISH feature map is the total energy of its
associated order. Mathematically, it is the sum of the squared SH
coefficients of the SH order.

This script supports both symmetrical and asymmetrical SH images as input, of
any SH order.

Each RISH feature will be saved as a separate file.

[1] Mirzaalian, Hengameh, et al. "Harmonizing diffusion MRI data across
multiple sites and scanners." MICCAI 2015.
https://scholar.harvard.edu/files/hengameh/files/miccai2015.pdf

positional arguments:
  in_sh         Path of the sh image. Must be a symmetric SH file.
  out_prefix    Prefix of the output RISH files to save.

optional arguments:
  -h, --help    show this help message and exit
  --full_basis  Input SH image uses a full SH basis (asymmetrical).
  --mask MASK   Path to a binary mask.
                Only data inside the mask will be used for computation.
  -f            Force overwriting of the output files.
```

2.56 scil_compute_seed_by_labels.py

```
usage: __main__.py [-h] [-f] in_labels in_labels_lut in_seed_maps

Computes the information from the seeding map for each cortical region
(corresponding to an atlas) associated with a specific bundle.
Here we want to estimate the seeding attribution to cortical area
affected by the bundle

positional arguments:
  in_labels      Path of the input label file.
```

(continues on next page)

(continued from previous page)

```
  in_labels_lut  Path of the LUT file corresponding to labels,used to name the␣
↪regions of interest.
  in_seed_maps   Path of the input seed map file.

optional arguments:
  -h, --help     show this help message and exit
  -f             Force overwriting of the output files.
```

2.57 scil_compute_seed_density_map.py

```
usage: __main__.py [-h] [--binary [FIXED_VALUE]] [-f] [--no_bbox_check]
                   tractogram_filename seed_density_filename

Compute a density map of seeds saved in .trk file.

positional arguments:
  tractogram_filename   Tracts filename. Format must be .trk.
                        File should contain a "seeds" value in the data_per_
↪streamline.
                        These seeds must be in space: voxel, origin: corner.
  seed_density_filename
                        Output seed density filename. Format must be Nifti.

optional arguments:
  -h, --help            show this help message and exit
  --binary [FIXED_VALUE]
                        If set, will store the same value for all intersected voxels,␣
↪creating a binary map.
                        When set without a value, 1 is used (and dtype uint8).
                        If a value is given, will be used as the stored value.
  -f                    Force overwriting of the output files.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).
```

2.58 scil_compute_sf_from_sh.py

```
usage: __main__.py [-h]
                   (--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724} | --in_bvec IN_BVEC)
                   [--dtype {float32,float64}] [--in_bval IN_BVAL]
                   [--in_b0 IN_B0] [--out_bval OUT_BVAL] [--out_bvec OUT_BVEC]
                   [--b0_scaling] [--sh_basis {descoteaux07,tournier07}]
                   [--full_basis] [--processes NBR] [-f]
                   [--force_b0_threshold]
                   in_sh out_sf

Script to sample SF values from a Spherical Harmonics signal. Outputs a Nifti
file with the SF values and an associated .bvec file with the chosen directions.

If converting from SH to a DWI-like SF volume, --in_bval and --in_b0 need
```

(continues on next page)

(continued from previous page)

```
to be provided to concatenate the b0 image to the SF, and to generate the new
bvals file. Otherwise, no .bval file will be created.

positional arguments:
  in_sh                 Path of the SH volume.
  out_sf                Name of the output SF file to save (bvals/bvecs will be␣
↪automatically named when necessary).

optional arguments:
  -h, --help            show this help message and exit
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Sphere used for the SH to SF projection.
  --in_bvec IN_BVEC     Directions used for the SH to SF projection.
  --dtype {float32,float64}
                        Datatype to use for SF computation and output array.'[float32]
↪'
  --in_bval IN_BVAL     b-value file, in FSL format, used to assign a b-value to the␣
↪output SF and generate a `.bval` file.
  --in_b0 IN_B0         b0 volume to concatenate to the final SF volume.
  --out_bval OUT_BVAL   Optional output bval file.
  --out_bvec OUT_BVEC   Optional output bvec file.
  --b0_scaling          Scale resulting SF by the b0 image.
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --full_basis          If true, use a full basis for the input SH coefficients.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
```

2.59 scil_compute_sh_from_signal.py

```
usage: __main__.py [-h] [--sh_order SH_ORDER]
                   [--sh_basis {descoteaux07,tournier07}] [--smooth SMOOTH]
                   [--use_attenuation] [--force_b0_threshold] [--mask MASK]
                   [-f]
                   in_dwi in_bval in_bvec out_sh

Script to compute the SH coefficient directly on the raw DWI signal.

positional arguments:
  in_dwi                Path of the dwi volume.
  in_bval               Path of the b-value file, in FSL format.
  in_bvec               Path of the b-vector file, in FSL format.
  out_sh                Name of the output SH file to save.

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help            show this help message and exit
  --sh_order SH_ORDER   SH order to fit (int). [4]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --smooth SMOOTH       Lambda-regularization coefficient in the SH fit (float). [0.
↪006]
  --use_attenuation     If set, will use signal attenuation before fitting the SH (i.
↪e. divide by the b0).
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  --mask MASK           Path to a binary mask.
                        Only data inside the mask will be used for computations and␣
↪reconstruction
  -f                    Force overwriting of the output files.
```

2.60 scil_compute_ssst_fodf.py

```
usage: __main__.py [-h] [--sh_order int] [--mask] [--force_b0_threshold]
                   [--sh_basis {descoteaux07,tournier07}] [--processes NBR]
                   [-f] [-v]
                   in_dwi in_bval in_bvec frf_file out_fODF

Script to compute Constrained Spherical Deconvolution (CSD) fiber ODFs.

See [Tournier et al. NeuroImage 2007]

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  in_bvec               Path of the bvecs file, in FSL format.
  frf_file              Path of the FRF file
  out_fODF              Output path for the fiber ODF coefficients.

optional arguments:
  -h, --help            show this help message and exit
  --sh_order int        SH order used for the CSD. (Default: 8)
  --mask                Path to a binary mask. Only the data inside the mask will be␣
↪used for computations and reconstruction.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
```

(continues on next page)

(continued from previous page)

```
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.61 scil_compute_ssst_frf.py

```
usage: __main__.py [-h] [--force_b0_threshold] [--mask MASK]
                   [--mask_wm MASK_WM] [--fa FA_THRESH]
                   [--min_fa MIN_FA_THRESH] [--min_nvox MIN_NVOX]
                   [--roi_radii ROI_RADII [ROI_RADII ...]]
                   [--roi_center tuple3) tuple(3) tuple(3] [-f] [-v]
                   in_dwi in_bval in_bvec frf_file

Compute a single Fiber Response Function from a DWI.

A DTI fit is made, and voxels containing a single fiber population are
found using a threshold on the FA.

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  in_bvec               Path of the bvecs file, in FSL format.
  frf_file              Path to the output FRF file, in .txt format, saved by
                        Numpy.

optional arguments:
  -h, --help            show this help message and exit
  --force_b0_threshold  If set, the script will continue even if the minimum
                        bvalue is suspiciously high ( > 20)
  --mask MASK           Path to a binary mask. Only the data inside the mask
                        will be used for computations and reconstruction.
                        Useful if no white matter mask is available.
  --mask_wm MASK_WM     Path to a binary white matter mask. Only the data
                        inside this mask and above the threshold defined by
                        --fa will be used to estimate the fiber response
                        function.
  --fa FA_THRESH        If supplied, use this threshold as the initial
                        threshold to select single fiber voxels. [0.7]
  --min_fa MIN_FA_THRESH
                        If supplied, this is the minimal value that will be
                        tried when looking for single fiber voxels. [0.5]
  --min_nvox MIN_NVOX   Minimal number of voxels needing to be identified as
                        single fiber voxels in the automatic estimation. [300]
  --roi_radii ROI_RADII [ROI_RADII ...]
                        If supplied, use those radii to select a cuboid roi to
                        estimate the response functions. The roi will be a
                        cuboid spanning from the middle of the volume in each
                        direction with the different radii. The type is either
                        an int or an array-like (3,). [[20]]
  --roi_center tuple(3) tuple(3) tuple(3)
                        If supplied, use this center to span the roi of size
                        roi_radius. [center of the 3D volume]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

(continues on next page)

(continued from previous page)

```
References: [1] Tournier et al. NeuroImage 2007
```

2.62 scil_compute_streamlines_density_map.py

```
usage: __main__.py [-h] [--binary [FIXED_VALUE]] [--reference REFERENCE] [-f]
                   in_bundle out_img

Compute a density map from a streamlines file.

A specific value can be assigned instead of using the tract count.

This script correctly handles compressed streamlines.

positional arguments:
  in_bundle             Tractogram filename. Format must be one of
                        trk, tck, vtk, fib, dpy.
  out_img               path of the output image file.

optional arguments:
  -h, --help            show this help message and exit
  --binary [FIXED_VALUE]
                        If set, will store the same value for all intersected voxels,␣
↪creating a binary map.
                        When set without a value, 1 is used (and dtype uint8).
                        If a value is given, will be used as the stored value.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.63 scil_compute_streamlines_length_stats.py

```
usage: __main__.py [-h] [--reference REFERENCE] [--indent INDENT]
                   [--sort_keys]
                   in_bundle

Compute streamlines min, mean and max length, as well as
standard deviation of length in mm.

positional arguments:
  in_bundle             Fiber bundle file.

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.64 scil_compute_todi.py

```
usage: __main__.py [-h] [--reference REFERENCE] [--sphere SPHERE]
                   [--mask MASK] [--out_mask OUT_MASK] [--out_tdi OUT_TDI]
                   [--out_todi_sf OUT_TODI_SF] [--out_todi_sh OUT_TODI_SH]
                   [--sh_order SH_ORDER] [--normalize_per_voxel]
                   [--smooth_todi] [--asymmetric] [--n_steps N_STEPS]
                   [--sh_basis {descoteaux07,tournier07}] [-f]
                   in_tractogram

Compute a Track Orientation Density Image (TODI).
Each segment of the streamlines is weighted by its length
(to support compressed streamlines).
This script can afterwards output a Track Density Image (TDI)
or a TODI with SF or SH representation, based on streamlines' segments.

positional arguments:
  in_tractogram         Input streamlines file.

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  --sphere SPHERE       sphere used for the angular discretization. [repulsion724]
  --mask MASK           Use the given mask.
  --out_mask OUT_MASK   Mask showing where TDI > 0.
  --out_tdi OUT_TDI     Output Track Density Image (TDI).
  --out_todi_sf OUT_TODI_SF
                        Output TODI, with SF (each directions
                        on the sphere, requires a lot of memory)
  --out_todi_sh OUT_TODI_SH
                        Output TODI, with SH coefficients.
  --sh_order SH_ORDER   Order of the original SH. [8]
  --normalize_per_voxel
                        Normalize each SF/SH at each voxel [False].
  --smooth_todi         Smooth TODI (angular and spatial) [False].
  --asymmetric          Compute asymmetric TODI [False].
  --n_steps N_STEPS     Number of steps for streamline segments subdivision prior to
↪binning [1].
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  -f                    Force overwriting of the output files.

    References:
        [1] Dhollander T, Emsell L, Van Hecke W, Maes F, Sunaert S, Suetens P.
            Track orientation density imaging (TODI) and
            track orientation distribution (TOD) based tractography.
            NeuroImage. 2014 Jul 1;94:312-36.
```

2.65 scil_concatenate_dwi.py

```
usage: __main__.py [-h] [--in_dwis IN_DWIS [IN_DWIS ...]]
                   [--in_bvals IN_BVALS [IN_BVALS ...]]
                   [--in_bvecs IN_BVECS [IN_BVECS ...]]
                   [--data_type DATA_TYPE] [-f]
                   out_dwi out_bval out_bvec

Concatenate DWI, bval and bvecs together. File must be specified in matching
order. Default data type will be the same as the first input DWI.

positional arguments:
  out_dwi               The name of the output DWI file.
  out_bval              The name of the output b-values file (.bval).
  out_bvec              The name of the output b-vectors file (.bvec).

optional arguments:
  -h, --help            show this help message and exit
  --in_dwis IN_DWIS [IN_DWIS ...]
                        The DWI file (.nii) to concatenate.
  --in_bvals IN_BVALS [IN_BVALS ...]
                        The b-values files in FSL format (.bval).
  --in_bvecs IN_BVECS [IN_BVECS ...]
                        The b-vectors files in FSL format (.bvec).
  --data_type DATA_TYPE
                        Data type of the output image. Use the format: uint8, int16,␣
↪int/float32, int/float64.
  -f                    Force overwriting of the output files.
```

2.66 scil_connectivity_math.py

```
usage: __main__.py [-h] [--data_type DATA_TYPE] [--exclude_background] [-f]
                   [-v]
                   {lower_threshold,upper_threshold,lower_threshold_eq,upper_
↪threshold_eq,lower_clip,upper_clip,absolute_value,round,ceil,floor,normalize_sum,
↪normalize_max,log_10,log_e,convert,invert,addition,subtraction,multiplication,
↪division,mean,std,union,intersection,difference}
                   in_matrices [in_matrices ...] out_matrix

Performs an operation on a list of matrices. The supported operations are
listed below.

Some operations such as multiplication or addition accept float value as
parameters instead of matrices.
> scil_connectivity_math.py multiplication mat.npy 10 mult_10.npy

    lower_threshold: MAT THRESHOLD
        All values below the threshold will be set to zero.
        All values above the threshold will be set to one.

    upper_threshold: MAT THRESHOLD
        All values below the threshold will be set to one.
        All values above the threshold will be set to zero.
        Equivalent to lower_threshold followed by an inversion.
```

(continues on next page)

(continued from previous page)

```
    lower_threshold_eq: MAT THRESHOLD
        All values below the threshold will be set to zero.
        All values above or equal the threshold will be set to one.

    upper_threshold_eq: MAT THRESHOLD
        All values below or equal the threshold will be set to one.
        All values above the threshold will be set to zero.
        Equivalent to lower_threshold followed by an inversion.

    lower_clip: MAT THRESHOLD
        All values below the threshold will be set to threshold.

    upper_clip: MAT THRESHOLD
        All values above the threshold will be set to threshold.

    absolute_value: MAT
        All negative values will become positive.

    round: MAT
        Round all decimal values to the closest integer.

    ceil: MAT
        Ceil all decimal values to the next integer.

    floor: MAT
        Floor all decimal values to the previous integer.

    normalize_sum: MAT
        Normalize the matrix so the sum of all values is one.

    normalize_max: MAT
        Normalize the matrix so the maximum value is one.

    log_10: MAT
        Apply a log (base 10) to all non zeros values of an matrix.

    log_e: MAT
        Apply a natural log to all non zeros values of an matrix.

    convert: MAT
        Perform no operation, but simply change the data type.

    invert: MAT
        Operation on binary matrix to interchange 0s and 1s in a binary mask.

    addition: MATs
        Add multiple matrices together.

    subtraction: MAT_1 MAT_2
        Subtract first matrix by the second (MAT_1 - MAT_2).

    multiplication: MATs
        Multiply multiple matrices together (danger of underflow and overflow)

    division: MAT_1 MAT_2
        Divide first matrix by the second (danger of underflow and overflow)
```

(continues on next page)

(continued from previous page)

```
        Ignore zeros values, excluded from the operation.

    mean: MATs
        Compute the mean of matrices.
        If a single 4D matrix is provided, average along the last dimension.

    std: MATs
        Compute the standard deviation average of multiple matrices.
        If a single 4D matrix is provided, compute the STD along the last
        dimension.

    union: MATs
        Operation on binary matrix to keep voxels, that are non-zero, in at
        least one file.

    intersection: MATs
        Operation on binary matrix to keep the voxels, that are non-zero,
        are present in all files.

    difference: MAT_1 MAT_2
        Operation on binary matrix to keep voxels from the first file that are
        not in the second file (non-zeros).


positional arguments:
  {lower_threshold,upper_threshold,lower_threshold_eq,upper_threshold_eq,lower_clip,
↪upper_clip,absolute_value,round,ceil,floor,normalize_sum,normalize_max,log_10,log_e,
↪convert,invert,addition,subtraction,multiplication,division,mean,std,union,
↪intersection,difference}
                        The type of operation to be performed on the matrices.
  in_matrices           The list of matrices files or parameters.
  out_matrix            Output matrix path.

optional arguments:
  -h, --help            show this help message and exit
  --data_type DATA_TYPE
                        Data type of the output image. Use the format: uint8, float16,
↪ int32.
  --exclude_background  Does not affect the background of the original matrices.
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.67 scil_convert_fdf.py

2.68 scil_convert_gradients_fsl_to_mrtrix.py

```
usage: __main__.py [-h] [-f] [-v] fsl_bval fsl_bvec mrtrix_enc

Script to convert bval/bvec FSL style to MRtrix style.

positional arguments:
  fsl_bval    Path to FSL b-value file (.bval).
  fsl_bvec    Path to FSL gradient directions file (.bvec).
```

(continues on next page)

(continued from previous page)

```
  mrtrix_enc  Path to gradient directions encoding file (.b).

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
  -v          If set, produces verbose output.
```

2.69 scil_convert_gradients_mrtrix_to_fsl.py

```
usage: __main__.py [-h] [-f] [-v] mrtrix_enc fsl_bval fsl_bvec

Script to convert bval/bvec MRtrix style to FSL style.

positional arguments:
  mrtrix_enc  Path to the gradient directions encoding file. (.b)
  fsl_bval    Path to output FSL b-value file (.bval).
  fsl_bvec    Path to output FSL gradient directions file (.bvec).

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
  -v          If set, produces verbose output.
```

2.70 scil_convert_json_to_xlsx.py

```
usage: __main__.py [-h] [--no_sort_subs] [--no_sort_bundles]
                   [--ignore_bundles FILE] [--stats_over_population] [-f]
                   in_json out_xlsx

 Convert a final aggregated json file to an Excel spreadsheet. Typically
used during the tractometry pipeline.

positional arguments:
  in_json               File containing the json stats (.json).
  out_xlsx              Output Excel file for the stats (.xlsx).

optional arguments:
  -h, --help            show this help message and exit
  --no_sort_subs        If set, subjects won't be sorted alphabetically.
  --no_sort_bundles     If set, bundles won't be sorted alphabetically.
  --ignore_bundles FILE
                        Path to a text file containing a list of bundles to ignore (.
↪txt).
                        One bundle, corresponding to keys in the json, per line.
  --stats_over_population
                        If set, consider the input stats to be over an entire␣
↪population and not subject-based.
  -f                    Force overwriting of the output files.
```

2.71 scil_convert_rgb.py

```
usage: __main__.py [-h] [-f] in_image out_image

Converts a RGB image encoded as a 4D image to a RGB image encoded as
a 3D image, or vice versa.

Typically, most software tools used in the SCIL (including MI-Brain) use
the former, while Trackvis uses the latter.

Input
-Case 1: 4D image where the 4th dimension contains 3 values.
-Case 2: 3D image, in Trackvis format where each voxel contains a
         tuple of 3 elements, one for each value.

Output
-Case 1: 3D image, in Trackvis format where each voxel contains a
         tuple of 3 elements, one for each value (uint8).
-Case 2: 4D image where the 4th dimension contains 3 values (uint8).

positional arguments:
  in_image    name of input RGB image.
              Either 4D or 3D image.
  out_image   name of output RGB image.
              Either 3D or 4D image.

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
```

2.72 scil_convert_sh_basis.py

```
usage: __main__.py [-h] [--processes NBR] [-f]
                   in_sh out_sh {descoteaux07,tournier07}

    Convert a SH file between the two commonly used bases
    ('descoteaux07' or 'tournier07'). The specified basis corresponds to the
    input data basis.

positional arguments:
  in_sh                 Input SH filename. (nii or nii.gz)
  out_sh                Output SH filename. (nii or nii.gz)
  {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.

optional arguments:
  -h, --help            show this help message and exit
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
```

(continues on next page)

(continued from previous page)

```
  -f                    Force overwriting of the output files.
```

2.73 scil_convert_surface.py

```
usage: __main__.py [-h] [-f] in_surface out_surface

Script to convert a surface (FreeSurfer or VTK supported).
    ".vtk", ".vtp", ".ply", ".stl", ".xml", ".obj"

> scil_convert_surface.py surf.vtk converted_surf.ply

positional arguments:
  in_surface   Input a surface (FreeSurfer or supported by VTK).
  out_surface  Output flipped surface (formats supported by VTK).

optional arguments:
  -h, --help   show this help message and exit
  -f           Force overwriting of the output files.

References:
[1] St-Onge, E., Daducci, A., Girard, G. and Descoteaux, M. 2018.
    Surface-enhanced tractography (SET). NeuroImage.
```

2.74 scil_convert_tensors.py

```
usage: __main__.py [-h] [-f] in_file out_file in_format out_format

Conversion of tensors (the 6 values from the triangular matrix) between various
software standards. We cannot discover the input format type, user must know
how the tensors were created.

    Dipy's order is [Dxx, Dxy, Dyy, Dxz, Dyz, Dzz]
       Shape: [i, j , k, 6].
       Ref: https://github.com/dipy/dipy/blob/master/dipy/reconst/dti.py#L1639

    MRTRIX's order is : [Dxx, Dyy, Dzz, Dxy, Dxz, Dyz]
       Shape: [i, j , k, 6].
       Ref: https://mrtrix.readthedocs.io/en/dev/reference/commands/dwi2tensor.html

    ANTS's order ('nifti format') is : [Dxx, Dxy, Dyy, Dxz, Dyz, Dzz].
       Shape: [i, j , k, 1, 6] (Careful, file is 5D).
       Ref: https://github.com/ANTsX/ANTs/wiki/Importing-diffusion-tensor-data-from-
↪other-software

    FSL's order is [Dxx, Dxy, Dxz, Dyy, Dyz, Dzz]
       Shape: [i, j , k, 6].
       Ref: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FDT/UserGuide
       (Also used for the Fibernavigator)


positional arguments:
```

(continues on next page)

(continued from previous page)

```
  in_file     Input tensors filename.
  out_file    Output tensors filename.
  in_format   Input format. Choices: ['fsl', 'nifti', 'mrtrix', 'dipy']
  out_format  Output format. Choices: ['fsl', 'nifti', 'mrtrix', 'dipy']

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
```

2.75 scil_convert_tractogram.py

```
usage: __main__.py [-h] [--reference REFERENCE] [-f] [--no_bbox_check]
                   IN_TRACTOGRAM OUTPUT_NAME

Conversion of '.tck', '.trk', '.fib', '.vtk' and 'dpy' files using updated file
format standard. TRK file always needs a reference file, a NIFTI, for
conversion. The FIB file format is in fact a VTK, MITK Diffusion supports it.

positional arguments:
  IN_TRACTOGRAM         Tractogram filename. Format must be one of
                        trk, tck, vtk, fib, dpy
  OUTPUT_NAME           Output filename. Format must be one of
                        trk, tck, vtk, fib, dpy

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).
```

2.76 scil_count_non_zero_voxels.py

```
usage: __main__.py [-h] [--out OUT_FILE] [--stats] [--id VALUE_ID] IN_FILE

Count the number of non-zero voxels in an image file.

If you give it an image with more than 3 dimensions, it will summarize the 4th
(or more) dimension to one voxel, and then find non-zero voxels over this.
This means that if there is at least one non-zero voxel in the 4th dimension,
this voxel of the 3D volume will be considered as non-zero.

positional arguments:
  IN_FILE         Input file name, in nifti format.

optional arguments:
  -h, --help      show this help message and exit
  --out OUT_FILE  name of the output file, which will be saved as a text file.
```

(continues on next page)

(continued from previous page)

```
  --stats           output the value using a stats format. Using this syntax will append
                    a line to the output file, instead of creating a file with only one␣
↪line.
                    This is useful to create a file to be used as the source of data␣
↪for a graph.
                    Can be combined with --id
  --id VALUE_ID     Id of the current count. If used, the value of this argument will be
                    output (followed by a ":") before the count value.
                    Mostly useful with --stats.
```

2.77 scil_count_streamlines.py

```
usage: __main__.py [-h] [--print_count_alone] [--indent INDENT] [--sort_keys]
                   in_tractogram

Return the number of streamlines in a tractogram. Only support trk and tck in
order to support the lazy loading from nibabel.

positional arguments:
  in_tractogram         Path of the input tractogram file.

optional arguments:
  -h, --help            show this help message and exit
  --print_count_alone   If true, prints the result only.
                        Else, prints the bundle name and count formatted as a json␣
↪dict.(default)

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.78 scil_crop_volume.py

```
usage: __main__.py [-h] [--ignore_voxel_size] [-f]
                   [--input_bbox INPUT_BBOX | --output_bbox OUTPUT_BBOX]
                   in_image out_image

Crop a volume using a given or an automatically computed bounding box. If a
previously computed bounding box file is given, the cropping will be applied
and the affine fixed accordingly.

Warning: This works well on masked images (like with FSL-Bet) volumes since
it's looking for non-zero data. Therefore, you should validate the results on
other types of images that haven't been masked.

positional arguments:
  in_image              Path of the nifti file to crop.
  out_image             Path of the cropped nifti file to write.

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --ignore_voxel_size   Ignore voxel size compatibility test between input bounding␣
↪box and data. Warning, use only if you know what you are doing.
  -f                    Force overwriting of the output files.
  --input_bbox INPUT_BBOX
                        Path of the pickle file from which to take the bounding box␣
↪to crop input file.
  --output_bbox OUTPUT_BBOX
                        Path of the pickle file where to write the computed bounding␣
↪box.
```

2.79 scil_cut_streamlines.py

```
usage: __main__.py [-h] [--resample STEP_SIZE] [--compress ERROR_RATE]
                   [--biggest_blob] [-f] [-v]
                   in_tractogram in_mask out_tractogram

Filters streamlines and only keeps the parts of streamlines within or
between the ROIs. The script accepts a single input mask, the mask has either
1 entity/blob or 2 entities/blobs (does not support disconnected voxels).
The option --biggest_blob can help if you have such a scenario.

The 1 entity scenario will 'trim' the streamlines so their longest segment is
within the bounding box or a binary mask.

The 2 entities scenario will cut streamlines so their segment are within the
bounding box or going from binary mask #1 to binary mask #2.

Both scenarios will erase data_per_point and data_per_streamline.

positional arguments:
  in_tractogram         Input tractogram file.
  in_mask               Binary mask containing either 1 or 2 blobs.
  out_tractogram        Output tractogram file.

optional arguments:
  -h, --help            show this help message and exit
  --resample STEP_SIZE  Resample streamlines to a specific step-size in mm [None].
  --compress ERROR_RATE
                        Maximum compression distance in mm [None].
  --biggest_blob        Use the biggest entity and force the 1 ROI scenario.
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.80 scil_decompose_connectivity.py

```
usage: __main__.py [-h] [--no_pruning] [--no_remove_loops]
                   [--no_remove_outliers] [--no_remove_curv_dev]
                   [--min_length MIN_LENGTH] [--max_length MAX_LENGTH]
                   [--outlier_threshold OUTLIER_THRESHOLD]
                   [--loop_max_angle LOOP_MAX_ANGLE]
                   [--curv_qb_distance CURV_QB_DISTANCE] [--out_dir OUT_DIR]
```

(continues on next page)

(continued from previous page)

```
                    [--save_raw_connections] [--save_intermediate]
                    [--save_discarded] [--out_labels_list OUT_FILE]
                    [--reference REFERENCE] [--processes NBR] [-v] [-f]
                    [--no_bbox_check]
                    in_tractogram in_labels out_hdf5

Compute a connectivity matrix from a tractogram and a parcellation.

Current strategy is to keep the longest streamline segment connecting
2 regions. If the streamline crosses other gray matter regions before
reaching its final connected region, the kept connection is still the
longest. This is robust to compressed streamlines.

The output file is a hdf5 (.h5) where the keys are 'LABEL1_LABEL2' and each
group is composed of 'data', 'offsets' and 'lengths' from the array_sequence.
The 'data' is stored in VOX/CORNER for simplicity and efficiency.

For the --outlier_threshold option the default is a recommended good trade-off
for a freesurfer parcellation. With smaller parcels (brainnetome, glasser) the
threshold should most likely be reduced.
Good candidate connections to QC are the brainstem to precentral gyrus
connection and precentral left to precentral right connection, or equivalent
in your parcellation."

NOTE: this script can take a while to run. Please be patient.
Example: on a tractogram with 1.8M streamlines, running on a SSD:
- 15 minutes without post-processing, only saving final bundles.
- 30 minutes with full post-processing, only saving final bundles.
- 60 minutes with full post-processing, saving all possible files.

positional arguments:
  in_tractogram         Tractogram filename. Format must be one of
                        trk, tck, vtk, fib, dpy.
  in_labels             Labels file name (nifti). Labels must have 0 as background.
  out_hdf5              Output hdf5 file (.h5).

optional arguments:
  -h, --help            show this help message and exit
  --out_labels_list OUT_FILE
                        Save the labels list as text file.
                        Needed for scil_compute_connectivity.py and others.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).

Post-processing options:
  --no_pruning          If set, will NOT prune on length.
                        Length criteria in --min_length, --max_length.
  --no_remove_loops     If set, will NOT remove streamlines making loops.
                        Angle criteria based on --loop_max_angle.
```

(continues on next page)

(continued from previous page)

```
  --no_remove_outliers  If set, will NOT remove outliers using QB.
                        Criteria based on --outlier_threshold.
  --no_remove_curv_dev  If set, will NOT remove streamlines that deviate from the␣
↪mean curvature.
                        Threshold based on --curv_qb_distance.

Pruning options:
  --min_length MIN_LENGTH
                        Pruning minimal segment length. [20.0]
  --max_length MAX_LENGTH
                        Pruning maximal segment length. [200.0]

Outliers and loops options:
  --outlier_threshold OUTLIER_THRESHOLD
                        Outlier removal threshold when using hierarchical QB. [0.6]
  --loop_max_angle LOOP_MAX_ANGLE
                        Maximal winding angle over which a streamline is considered␣
↪as looping. [330.0]
  --curv_qb_distance CURV_QB_DISTANCE
                        Clustering threshold for centroids curvature filtering with␣
↪QB. [10.0]

Saving options:
  --out_dir OUT_DIR     Output directory for each connection as separate file (.trk).
  --save_raw_connections
                        If set, will save all raw cut connections in a subdirectory.
  --save_intermediate   If set, will save the intermediate results of filtering.
  --save_discarded      If set, will save discarded streamlines in subdirectories.
                        Includes loops, outliers and qb_loops.
```

2.81 scil_detect_dwi_volume_outliers.py

```
usage: __main__.py [-h] [--b0_thr B0_THR] [--std_scale STD_SCALE]
                   [--force_b0_threshold] [-v]
                   in_dwi in_bval in_bvec

This script simply finds the 3 closest angular neighbors of each direction
(per shell) and compute the voxel-wise correlation.
If the angles or correlations to neighbors are below the shell average (by
args.std_scale x STD) it will flag the volume as a potential outlier.

This script supports multi-shells, but each shell is independant and detected
using the args.b0_thr parameter.

This script can be run before any processing to identify potential problem
before launching pre-processing.

positional arguments:
  in_dwi                The DWI file (.nii) to concatenate.
  in_bval               The b-values files in FSL format (.bval).
  in_bvec               The b-vectors files in FSL format (.bvec).

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --b0_thr B0_THR       All b-values with values less than or equal to b0_thr are␣
↪considered as b0s i.e. without diffusion weighting. [20.0]
  --std_scale STD_SCALE
                        How many deviation from the mean are required to be␣
↪considered an outliers. [2.0]
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  -v                    If set, produces verbose output.
```

2.82 scil_detect_streamlines_loops.py

```
usage: __main__.py [-h] [--looping_tractogram LOOPING_TRACTOGRAM] [--qb]
                   [--threshold THRESHOLD] [-a ANGLE] [--display_counts]
                   [--processes NBR] [-f] [--reference REFERENCE]
                   [--indent INDENT] [--sort_keys]
                   in_tractogram out_tractogram

This script can be used to remove loops in two types of streamline datasets:

  - Whole brain: For this type, the script removes streamlines if they
    make a loop with an angle of more than 360 degrees. It's possible to change
    this angle with the -a option. Warning: Don't use --qb option for a
    whole brain tractography.

  - Bundle dataset: For this type, it is possible to remove loops and
    streamlines outside of the bundle. For the sharp angle turn, use --qb
    option.

------------------------------------------------------------------------------
Reference:
QuickBundles based on [Garyfallidis12] Frontiers in Neuroscience, 2012.
------------------------------------------------------------------------------

positional arguments:
  in_tractogram         Tractogram input file name.
  out_tractogram        Output tractogram without loops.

optional arguments:
  -h, --help            show this help message and exit
  --looping_tractogram LOOPING_TRACTOGRAM
                        If set, saves detected looping streamlines.
  --qb                  If set, uses QuickBundles to detect
                        outliers (loops, sharp angle turns).
                        Should mainly be used with bundles. [False]
  --threshold THRESHOLD
                        Maximal streamline to bundle distance
                        for a streamline to be considered as
                        a tracking error. [8.0]
  -a ANGLE              Maximum looping (or turning) angle of
                        a streamline in degrees. [360]
  --display_counts      Print streamline count before and after filtering
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
```

(continues on next page)

(continued from previous page)

```
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.83 scil_dilate_labels.py

```
usage: __main__.py [-h] [--distance DISTANCE]
                   [--labels_to_dilate LABELS_TO_DILATE [LABELS_TO_DILATE ...]]
                   [--labels_to_fill LABELS_TO_FILL [LABELS_TO_FILL ...]]
                   [--labels_not_to_dilate LABELS_NOT_TO_DILATE [LABELS_NOT_TO_DILATE␣
↪...]]
                   [--mask MASK] [--processes NBR] [-f]
                   in_file out_file

Dilate regions (with or without masking) from a labeled volume:
- "label_to_dilate" are regions that will dilate over
    "label_to_fill" if close enough to it ("distance").
- "label_to_dilate", by default (None) will be all
       non-"label_to_fill" and non-"label_not_to_dilate".
- "label_not_to_dilate" will not be changed, but will not dilate.
- "mask" is where the dilation is allowed (constrained)
    in addition to "background_label" (logical AND)

>>> scil_dilate_labels.py wmparc_t1.nii.gz wmparc_dil.nii.gz \
    --label_to_fill 0 5001 5002 \
    --label_not_to_dilate 4 43 10 11 12 49 50 51

positional arguments:
  in_file               Path of the volume (nii or nii.gz).
  out_file              Output filename of the dilated labels.

optional arguments:
  -h, --help            show this help message and exit
  --distance DISTANCE   Maximal distance to dilate (in mm) [2.0].
  --labels_to_dilate LABELS_TO_DILATE [LABELS_TO_DILATE ...]
                        Label list to dilate. By default it dilates all
                        labels not in labels_to_fill nor in labels_not_to_dilate.
  --labels_to_fill LABELS_TO_FILL [LABELS_TO_FILL ...]
                        Background id / labels to be filled [[0]],
                         the first one is given as output background value.
  --labels_not_to_dilate LABELS_NOT_TO_DILATE [LABELS_NOT_TO_DILATE ...]
                        Label list not to dilate.
  --mask MASK           Only dilate values inside the mask.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.

    References:
        [1] Al-Sharif N.B., St-Onge E., Vogel J.W., Theaud G.,
            Evans A.C. and Descoteaux M. OHBM 2019.
```

(continues on next page)

(continued from previous page)

```
            Surface integration for connectome analysis in age prediction.
```

2.84 scil_estimate_bundles_diameter.py

```
usage: __main__.py [-h] [--fitting_func {lin_up,lin_down,exp,inv,log}]
                   [--show_rendering | --save_rendering OUT_FOLDER]
                   [--wireframe] [--error_coloring] [--width WIDTH]
                   [--opacity OPACITY] [--background R G B]
                   [--reference REFERENCE] [--indent INDENT] [--sort_keys]
                   [-f]
                   in_bundles [in_bundles ...] in_labels [in_labels ...]

Script to estimate the diameter of bundle(s) along their length.
The script expects:
- bundles with coherent endpoints from scil_uniformize_streamlines_endpoints.py
- labels maps with around 5-50 points scil_compute_bundle_voxel_label_map.py
    <5 is not enough, high risk of bad fit
    >50 is too much, high risk of bad fit
- bundles that are close to a tube
    without major fanning in a single axis
    fanning is in 2 directions (uniform dispersion) good approximation

The scripts prints a JSON file with mean/std to be compatible with tractometry.
WARNING: STD is in fact an ERROR measure from the fit and NOT an STD.

Since the estimation and fit quality is not always intuitive for some bundles
and the tube with varying diameter is not easy to color/visualize,
the script comes with its own VTK rendering to allow exploration of the data.
(optional).

positional arguments:
  in_bundles            List of tractography files supported by nibabel.
  in_labels             List of labels maps that matches the bundles.

optional arguments:
  -h, --help            show this help message and exit
  --fitting_func {lin_up,lin_down,exp,inv,log}
                        Function to weigh points using their distance.
                        [Default: None]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Visualization options:
  --show_rendering      Display VTK window (optional).
  --save_rendering OUT_FOLDER
                        Save VTK render in the specified folder (optional)
  --wireframe           Use wireframe for the tube rendering.
  --error_coloring      Use the fitting error to color the tube.
  --width WIDTH         Width of tubes or lines representing streamlines
                        [Default: 0.2]
  --opacity OPACITY     Opacity for the streamlines rendered with the tube.
                        [Default: 0.2]
```

(continues on next page)

(continued from previous page)

```
  --background R G B    RBG values [0, 255] of the color of the background.
                        [Default: [1, 1, 1]]

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.85 scil_evaluate_bundles_binary_classification_measures.py

```
usage: __main__.py [-h]
                   [--streamlines_measures GOLD_STANDARD_STREAMLINES TRACTOGRAM]
                   [--voxels_measures GOLD_STANDARD_MASK TRACKING MASK]
                   [--processes NBR] [--reference REFERENCE] [-v]
                   [--indent INDENT] [--sort_keys] [-f]
                   in_bundles [in_bundles ...] out_json

Evaluate binary classification measures between gold standard and bundles.
All tractograms must be in the same space (aligned to one reference)
The measures can be applied to voxel-wise or streamline-wise representation.

A gold standard must be provided for the desired representation.
A gold standard would be a segmentation from an expert or a group of experts.
If only the streamline-wise representation is provided without a voxel-wise
gold standard, it will be computed from the provided streamlines.
At least one of the two representations is required.

The gold standard tractogram is the tractogram (whole brain most likely) from
which the segmentation is performed.
The gold standard tracking mask is the tracking mask used by the tractography
algorighm to generate the gold standard tractogram.

The computed binary classification measures are:
sensitivity, specificity, precision, accuracy, dice, kappa, youden for both
the streamline and voxel representation (if provided).

positional arguments:
  in_bundles            Path of the input bundles.
  out_json              Path of the output json.

optional arguments:
  -h, --help            show this help message and exit
  --streamlines_measures GOLD_STANDARD_STREAMLINES TRACTOGRAM
                        The gold standard bundle and the original tractogram.
  --voxels_measures GOLD_STANDARD_MASK TRACKING MASK
                        The gold standard mask and the original tracking mask.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Json options:
```

(continues on next page)

(continued from previous page)

```
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.86 scil_evaluate_bundles_individual_measures.py

```
usage: __main__.py [-h] [--group_statistics] [--no_uniformize]
                   [--reference REFERENCE] [--processes NBR] [--indent INDENT]
                   [--sort_keys] [-f]
                   in_bundles [in_bundles ...] out_json

Evaluate basic measurements of bundles, all at once.
All tractograms must be in the same space (aligned to one reference)
The computed measures are:
volume, volume_endpoints, streamlines_count, avg_length, std_length,
min_length, max_length, span, curl, diameter, elongation, surface area,
irregularity, end surface area, radius, end surface irregularity,
mean_curvature, fractal dimension.

The set average contains the average measures of all input bundles. The
measures that are dependent on the streamline count are weighted by the number
of streamlines of each bundle. Each of these average measure is computed by
first summing the multiple of a measure and the streamline count of each
bundle and divide the sum by the total number of streamlines. Thus, measures
including length and span are essentially averages of all the streamlines.
Other streamline-related set measure are computed with other set averages.
Whereas, bundle-related measures are computed as an average of all bundles.
These measures include volume and surface area.

The fractal dimension is dependent on the voxel size and the number of voxels.
If data comparison is performed, the bundles MUST be in same resolution.

positional arguments:
  in_bundles            Path of the input bundles.
  out_json              Path of the output file.

optional arguments:
  -h, --help            show this help message and exit
  --group_statistics    Show average measures
                        [False].
  --no_uniformize       Do NOT automatically uniformize endpoints for theendpoints␣
↪related metrics.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.

References:
[1] Fang-Cheng Yeh. 2020.
```

(continues on next page)

(continued from previous page)

```
    Shape analysis of the human association pathways. NeuroImage.
```

2.87 scil_evaluate_bundles_pairwise_agreement_measures.py

```
usage: __main__.py [-h] [--streamline_dice] [--bundle_adjency_no_overlap]
                   [--disable_streamline_distance]
                   [--single_compare SINGLE_COMPARE] [--keep_tmp] [--ratio]
                   [--processes NBR] [--reference REFERENCE] [--indent INDENT]
                   [--sort_keys] [-f]
                   in_bundles [in_bundles ...] out_json

Evaluate pair-wise similarity measures of bundles.
All tractograms must be in the same space (aligned to one reference)

For the voxel representation the computed similarity measures are:
bundle_adjacency_voxels, dice_voxels, w_dice_voxels, density_correlation
volume_overlap, volume_overreach
The same measures are also evluated for the endpoints.

For the streamline representation the computed similarity measures are:
bundle_adjacency_streamlines, dice_streamlines, streamlines_count_overlap,
streamlines_count_overreach

positional arguments:
  in_bundles            Path of the input bundles.
  out_json              Path of the output json file.

optional arguments:
  -h, --help            show this help message and exit
  --streamline_dice     Compute streamline-wise dice coefficient.
                        Tractograms must be identical [False].
  --bundle_adjency_no_overlap
                        If set, do not count zeros in the average BA.
  --disable_streamline_distance
                        Will not compute the streamlines distance
                        [False].
  --single_compare SINGLE_COMPARE
                        Compare inputs to this single file.
  --keep_tmp            Will not delete the tmp folder at the end.
  --ratio               Compute overlap and overreach as a ratio over the
                        reference tractogram in a Tractometer-style way.
                        Can only be used if also using the `single_compare` option.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.88 scil_evaluate_connectivity_graph_measures.py

```
usage: __main__.py [-h] [--filtering_mask FILTERING_MASK] [--avg_node_wise]
                   [--append_json] [--small_world] [--indent INDENT]
                   [--sort_keys] [-v] [-f]
                   in_conn_matrix in_length_matrix out_json

Evaluate graph theory measures from connectivity matrices.
A length weighted and a streamline count weighted matrix are required since
some measures require one or the other.

This script evaluates the measures one subject at the time. To generate a
population dictionary (similarly to other scil_evaluate_*.py scripts), use the
--append_json option as well as using the same output filename.
>>> for i in hcp/*/; do scil_evaluate_connectivity_measures.py ${i}/sc_prob.npy
    ${i}/len_prob.npy hcp_prob.json --append_json --avg_node_wise; done

Some measures output one value per node, the default behavior is to list
them all into a list. To obtain only the average use the
--avg_node_wise option.

The computed connectivity measures are:
centrality, modularity, assortativity, participation, clustering,
nodal_strength, local_efficiency, global_efficiency, density, rich_club,
path_length, edge_count, omega, sigma

For more details about the measures, please refer to
- https://sites.google.com/site/bctnet/measures
- https://github.com/aestrivex/bctpy/wiki

This script is under the GNU GPLv3 license, for more detail please refer to
https://www.gnu.org/licenses/gpl-3.0.en.html

positional arguments:
  in_conn_matrix        Input connectivity matrix (.npy).
                        Typically a streamline count weighted matrix.
  in_length_matrix      Input length weighted matrix (.npy).
  out_json              Path of the output json.

optional arguments:
  -h, --help            show this help message and exit
  --filtering_mask FILTERING_MASK
                        Binary filtering mask to apply before computing the measures.
  --avg_node_wise       Return a single value for node-wise measures.
  --append_json         If the file already exists, will append to the dictionary.
  --small_world         Compute measure related to small worldness (omega and sigma).
                         This option is much slower.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.

[1] Rubinov, Mikail, and Olaf Sporns. "Complex network measures of brain
    connectivity: uses and interpretations." Neuroimage 52.3 (2010):
    1059-1069.
```

2.89 scil_evaluate_connectivity_pairwise_agreement_measures.py

```
usage: __main__.py [-h] [--single_compare SINGLE_COMPARE] [--normalize]
                   [--indent INDENT] [--sort_keys] [-f]
                   in_matrices [in_matrices ...] out_json

Evaluate pair-wise similarity measures of connectivity matrix.

The computed similarity measures are:
sum of square difference and pearson correlation coefficent

positional arguments:
  in_matrices           Path of the input matricies.
  out_json              Path of the output json file.

optional arguments:
  -h, --help            show this help message and exit
  --single_compare SINGLE_COMPARE
                        Compare inputs to this single file.
  --normalize           If set, will normalize all matrices from zero to one.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.90 scil_execute_angle_aware_bilateral_filtering.py

```
usage: __main__.py [-h] [--sh_basis {descoteaux07,tournier07}]
                   [--out_sym OUT_SYM]
                   [--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724}]
                   [--sigma_angular SIGMA_ANGULAR]
                   [--sigma_spatial SIGMA_SPATIAL] [--sigma_range SIGMA_RANGE]
                   [--use_gpu] [-v] [-f] [--processes NBR]
                   in_sh out_sh

Script to compute angle-aware bilateral filtering.

Angle-aware bilateral filtering is an extension of bilateral filtering
considering the angular distance between sphere directions for filtering
5-dimensional spatio-angular images.

The filtering can be performed on the GPU using pyopencl by specifying
--use_gpu. Make sure you have pyopencl installed to use this option.
Otherwise, the filtering also runs entirely on the CPU, optionally using
multiple processes.

Using default parameters, fODF filtering for a HCP subject processed with
Tractoflow takes about 12 minutes on the GPU versus 90 minutes using 16 CPU
threads. The time required scales with the sigma_spatial parameter. For
example, sigma_spatial=3.0 takes about 4.15 hours on the GPU versus 7.67 hours
on the CPU using 16 threads.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_sh                 Path to the input file.
  out_sh                File name for averaged signal.

optional arguments:
  -h, --help            show this help message and exit
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --out_sym OUT_SYM     Name of optional symmetric output. [None]
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Sphere used for the SH to SF projection. [repulsion724]
  --sigma_angular SIGMA_ANGULAR
                        Standard deviation for angular distance. [1.0]
  --sigma_spatial SIGMA_SPATIAL
                        Standard deviation for spatial distance. [1.0]
  --sigma_range SIGMA_RANGE
                        Standard deviation for range filter. [1.0]
  --use_gpu             Use GPU for computation.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
```

2.91 scil_execute_asymmetric_filtering.py

```
usage: __main__.py [-h] [--out_sym OUT_SYM]
                   [--sh_basis {descoteaux07,tournier07}]
                   [--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724}]
                   [--sharpness SHARPNESS] [--sigma SIGMA] [-v] [-f]
                   in_sh out_sh

Script to compute per-vertices hemisphere-aware (asymmetric)
filtering of spherical functions (SF) given an array of spherical harmonics
(SH) coefficients. SF are filtered using a first-neighbor Gaussian filter.
Sphere directions are also weighted by their dot product with the direction
to the center of each neighbor, clipping to 0 negative values.

The argument `sigma` controls the standard deviation of the Gaussian. The
argument `sharpness` controls the exponent of the cosine weights. The higher it
is, the faster the weights of misaligned sphere directions decrease. A
sharpness of 0 gives the same weight to all sphere directions in an hemisphere.
Both `sharpness` and `sigma` must be positive.

The resulting SF can be expressed using a full SH basis or a symmetric SH basis
(where the effect of the filtering is a simple denoising).

Using default parameters, the script completes in about 15-20 minutes for a
```

(continues on next page)

(continued from previous page)

```
HCP subject fiber ODF processed with tractoflow. Also note the bigger the
sphere used for SH to SF projection, the higher the RAM consumption and
compute time.

positional arguments:
  in_sh                 Path to the input file.
  out_sh                File name for averaged signal.

optional arguments:
  -h, --help            show this help message and exit
  --out_sym OUT_SYM     Name of optional symmetric output. [None]
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Sphere used for the SH to SF projection. [repulsion724]
  --sharpness SHARPNESS
                        Specify sharpness factor to use for weighted average. [1.0]
  --sigma SIGMA         Sigma of the gaussian to use. [1.0]
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.92 scil_extract_b0.py

```
usage: __main__.py [-h] [--b0_thr B0_THR]
                   [--all | --mean | --cluster-mean | --cluster-first]
                   [--block-size INT] [--single-image] [--force_b0_threshold]
                   [-v]
                   in_dwi in_bval in_bvec out_b0

Extract B0s from DWI.

The default behavior is to save the first b0 of the series.

positional arguments:
  in_dwi                DWI Nifti image.
  in_bval               b-values filename, in FSL format (.bval).
  in_bvec               b-values filename, in FSL format (.bvec).
  out_b0                Output b0 file(s).

optional arguments:
  -h, --help            show this help message and exit
  --b0_thr B0_THR       All b-values with values less than or equal to b0_thr are␣
↪considered as b0s i.e. without diffusion weighting. [0.0]
  --all                 Extract all b0. Index number will be appended to the output␣
↪file.
  --mean                Extract mean b0.
  --cluster-mean        Extract mean of each continuous cluster of b0s.
  --cluster-first       Extract first b0 of each continuous cluster of b0s.
```

(continues on next page)

(continued from previous page)

```
  --block-size INT, -s INT
                        Load the data using this block size. Useful
                        when the data is too large to be loaded in memory.
  --single-image        If output b0 volume has multiple time points, only outputs a␣
↪single image instead of a numbered series of images.
  --force_b0_threshold  If set, the script will continue even if the minimum bvalue␣
↪is suspiciously high ( > 20)
  -v                    If set, produces verbose output.
```

2.93 scil_extract_dwi_shell.py

```
usage: __main__.py [-h] [--block-size INT] [--tolerance INT] [-v] [-f]
                   in_dwi in_bval in_bvec in_bvals_to_extract
                   [in_bvals_to_extract ...] out_dwi out_bval out_bvec

Extracts the DWI volumes that are on specific b-value shells. Many shells
can be extracted at once by specifying multiple b-values. The extracted
volumes are in the same order as in the original file.

If the b-values of a shell are not all identical, use the --tolerance
argument to adjust the accepted interval. For example, a b-value of 2000
and a tolerance of 20 will extract all volumes with a b-values from 1980 to
2020.

Files that are too large to be loaded in memory can still be processed by
setting the --block-size argument. A block size of X means that X DWI volumes
are loaded at a time for processing.

positional arguments:
  in_dwi                The DW image file to split.
  in_bval               The b-values file in FSL format (.bval).
  in_bvec               The b-vectors file in FSL format (.bvec).
  in_bvals_to_extract   The list of b-values to extract. For example 0 2000.
  out_dwi               The name of the output DWI file.
  out_bval              The name of the output b-value file (.bval).
  out_bvec              The name of the output b-vector file (.bvec).

optional arguments:
  -h, --help            show this help message and exit
  --block-size INT, -s INT
                        Loads the data using this block size. Useful
                        when the data is too large to be loaded in memory.
  --tolerance INT, -t INT
                        The tolerated gap between the b-values to extract
                        and the actual b-values.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.94 scil_extract_ushape.py

```
usage: __main__.py [-h] [--minU MINU] [--maxU MAXU]
                   [--remaining_tractogram REMAINING_TRACTOGRAM] [--no_empty]
                   [--display_counts] [-f] [--reference REFERENCE]
                   [--indent INDENT] [--sort_keys]
                   in_tractogram out_tractogram

This script extracts streamlines depending on their U-shapeness.
This script is a replica of Trackvis method.

When ufactor is close to:
*  0 it defines straight streamlines
*  1 it defines U-fibers
* -1 it defines S-fibers

positional arguments:
  in_tractogram         Tractogram input file name.
  out_tractogram        Output tractogram file name.

optional arguments:
  -h, --help            show this help message and exit
  --minU MINU           Min ufactor value. [0.5]
  --maxU MAXU           Max ufactor value. [1.0]
  --remaining_tractogram REMAINING_TRACTOGRAM
                        If set, saves remaining streamlines.
  --no_empty            Do not write file if there is no streamline.
  --display_counts      Print streamline count before and after filtering.
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.95 scil_filter_connectivity.py

```
usage: __main__.py [-h] [--lower_than [LOWER_THAN [LOWER_THAN ...]]]
                   [--greater_than [GREATER_THAN [GREATER_THAN ...]]]
                   [--keep_condition_count] [--inverse_mask] [-v] [-f]
                   out_matrix_mask

Script to facilitate filtering of connectivity matrices.
The same could be achieved through a complex sequence of scil_connectivity_math.py.

Can be used with any connectivity matrix from scil_compute_connectivity.py.

For example, a simple filtering (Jasmeen style) would be:
scil_filter_connectivity.py out_mask.npy
    --greater_than */sc.npy 1 0.90
    --lower_than */sim.npy 2 0.90
    --greater_than */len.npy 40 0.90 -v;
```

(continues on next page)

(continued from previous page)

```
This will result in a binary mask where each node with a value of 1 represents
a node with at least 90% of the population having at least 1 streamline,
90% of the population is similar to the average (2mm) and 90% of the
population having at least 40mm of average streamlines length.

All operation are stricly > or <, there is no >= or <=.

--greater_than or --lower_than expect the same convention:
    MATRICES_LIST VALUE_THR POPULATION_PERC
It is strongly recommended (but not enforced) that the same number of
connectivity matrices is used for each condition.

This script performs an intersection of all conditions, meaning that all
conditions must be met in order not to be filtered.
If the user wants to manually handle the requirements, --keep_condition_count
can be used and manually binarized using scil_connectivity_math.py

positional arguments:
  out_matrix_mask       Output mask (matrix) resulting from the provided conditions (.
↪npy).

optional arguments:
  -h, --help            show this help message and exit
  --lower_than [LOWER_THAN [LOWER_THAN ...]]
                        Lower than condition using the VALUE_THR in at least␣
↪POPULATION_PERC (from MATRICES_LIST).
                        See description for more details.
  --greater_than [GREATER_THAN [GREATER_THAN ...]]
                        Greater than condition using the VALUE_THR in at least␣
↪POPULATION_PERC (from MATRICES_LIST).
                        See description for more details.
  --keep_condition_count
                        Report the number of condition(s) that pass/fail rather than␣
↪a binary mask.
  --inverse_mask        Inverse the final mask. 0 where all conditions are respected␣
↪and 1 where at least one fail.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.96 scil_filter_streamlines_by_length.py

```
usage: __main__.py [-h] [--minL MINL] [--maxL MAXL] [--no_empty]
                   [--display_counts] [--reference REFERENCE] [-f] [-v]
                   [--indent INDENT] [--sort_keys]
                   in_tractogram out_tractogram

Script to filter streamlines based on their lengths.

positional arguments:
  in_tractogram         Streamlines input file name.
  out_tractogram        Streamlines output file name.

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help            show this help message and exit
  --minL MINL           Minimum length of streamlines, in mm. [0.0]
  --maxL MAXL           Maximum length of streamlines, in mm. [inf]
  --no_empty            Do not write file if there is no streamline.
  --display_counts      Print streamline count before and after filtering
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.97 scil_filter_streamlines_by_orientation.py

```
usage: __main__.py [-h] [--min_x MIN_X] [--max_x MAX_X] [--min_y MIN_Y]
                   [--max_y MAX_Y] [--min_z MIN_Z] [--max_z MAX_Z] [--use_abs]
                   [--no_empty] [--display_counts] [--save_rejected filename]
                   [--reference REFERENCE] [-f] [-v] [--indent INDENT]
                   [--sort_keys]
                   in_tractogram out_tractogram

Script to filter streamlines based on their distance traveled in a specific
dimension (x, y, or z).

Useful to help differentiate bundles.

Examples: In a brain aligned with x coordinates in left - right axis and y
coordinates in anterior-posterior axis, a streamline from the ...
    - corpus callosum will likely travel a very short distance in the y axis.
    - cingulum will likely travel a very short distance in the x axis.

Note: we consider that x, y, z are the coordinates of the streamlines; we
do not verify if they are aligned with the brain's orientation.

positional arguments:
  in_tractogram         Streamlines input file name.
  out_tractogram        Streamlines output file name.

optional arguments:
  -h, --help            show this help message and exit
  --min_x MIN_X         Minimum distance in the first dimension, in mm.[0.0]
  --max_x MAX_X         Maximum distance in the first dimension, in mm.[inf]
  --min_y MIN_Y         Minimum distance in the second dimension, in mm.[0.0]
  --max_y MAX_Y         Maximum distance in the second dimension, in mm.[inf]
  --min_z MIN_Z         Minimum distance in the third dimension, in mm.[0.0]
  --max_z MAX_Z         Maximum distance in the third dimension, in mm.[inf]
  --use_abs             If set, will use the total of distances in absolute value (ex,
↪ coming back on yourself will contribute to the total distance instead of␣
↪cancelling it).
  --no_empty            Do not write file if there is no streamline.
  --display_counts      Print streamline count before and after filtering.
```

(continues on next page)

(continued from previous page)

```
  --save_rejected filename
                        Save the SFT of rejected streamlines.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.98 scil_filter_tractogram.py

```
usage: __main__.py [-h] [--drawn_roi ROI_NAME MODE CRITERIA]
                   [--atlas_roi ROI_NAME ID MODE CRITERIA]
                   [--bdo BDO_NAME MODE CRITERIA]
                   [--x_plane PLANE MODE CRITERIA]
                   [--y_plane PLANE MODE CRITERIA]
                   [--z_plane PLANE MODE CRITERIA]
                   [--filtering_list FILTERING_LIST]
                   [--soft_distance SOFT_DISTANCE] [--extract_masks_atlas_roi]
                   [--no_empty] [--display_counts] [--save_rejected FILENAME]
                   [--reference REFERENCE] [-v] [-f] [--indent INDENT]
                   [--sort_keys]
                   in_tractogram out_tractogram

Now supports sequential filtering condition and mixed filtering object.
For example, --atlas_roi ROI_NAME ID MODE CRITERIA
- ROI_NAME is the filename of a Nifti
- ID is one or multiple integer values in the atlas. If multiple values,
    ID needs to be between quotes.
    Example: "1:6 9 10:15" will use values between 1 and 6 and values
                           between 10 and 15 included as well as value 9.
- MODE must be one of these values: ['any', 'all', 'either_end', 'both_ends']
- CRITERIA must be one of these values: ['include', 'exclude']

If any meant any part of the streamline must be in the mask, all means that
all part of the streamline must be in the mask.

When used with exclude, it means that a streamline entirely in the mask will
be excluded. Using all it with x/y/z plane works but makes very little sense.

In terms of nifti mask, --drawn_roi MASK.nii.gz all include is
equivalent to --drawn_roi INVERSE_MASK.nii.gz any exclude
For example, this allows to find out all streamlines entirely in the WM in
one command (without manually inverting the mask first) or
to remove any streamlines staying in GM without getting out.

Multiple filtering tuples can be used and options mixed.
A logical AND is the only behavior available. All theses filtering
conditions will be sequentially applied.

WARNING: --soft_distance should be used carefully with large voxel size
```

(continues on next page)

(continued from previous page)

```
(e.g > 2.5mm).

positional arguments:
  in_tractogram         Path of the input tractogram file.
  out_tractogram        Path of the output tractogram file.

optional arguments:
  -h, --help            show this help message and exit
  --drawn_roi ROI_NAME MODE CRITERIA
                        Filename of a hand drawn ROI (.nii or .nii.gz).
  --atlas_roi ROI_NAME ID MODE CRITERIA
                        Filename of an atlas (.nii or .nii.gz).
  --bdo BDO_NAME MODE CRITERIA
                        Filename of a bounding box (bdo) file from MI-Brain.
  --x_plane PLANE MODE CRITERIA
                        Slice number in X, in voxel space.
  --y_plane PLANE MODE CRITERIA
                        Slice number in Y, in voxel space.
  --z_plane PLANE MODE CRITERIA
                        Slice number in Z, in voxel space.
  --filtering_list FILTERING_LIST
                        Text file containing one rule per line
                        (i.e. drawn_roi mask.nii.gz both_ends include).
  --soft_distance SOFT_DISTANCE
                        All ROIs are enlarged by the specified value.
                        The value is in voxel (NOT mm).
                        Anisotropic data will affect each direction differently
  --extract_masks_atlas_roi
                        Extract atlas roi masks.
  --no_empty            Do not write file if there is no streamline.
  --display_counts      Print streamline count before and after filtering
  --save_rejected FILENAME
                        Save rejected streamlines to output tractogram.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.99 scil_filter_tractogram_anatomically.py

```
usage: __main__.py [-h] [--minL MINL] [--maxL MAXL] [-a ANGLE]
                   [--csf_bin CSF_BIN]
                   [--ctx_dilation_radius CTX_DILATION_RADIUS]
                   [--save_intermediate_tractograms] [--save_volumes]
                   [--save_counts] [--no_empty] [--processes NBR]
                   [--reference REFERENCE] [-v] [-f] [--indent INDENT]
                   [--sort_keys]
                   in_tractogram in_wmparc out_path
```

(continues on next page)

(continued from previous page)

```
This script filters streamlines in a tractogram according to their geometrical
properties (i.e. limiting their length and looping angle) and their anatomical
ending properties (i.e. the anatomical tissue or region their endpoints lie
in). The filtering is performed sequentially in four steps, each step
processing the data on the output of the previous step:

    Step 1 - Remove streamlines below the minimum length and above the
             maximum length. These thresholds must be set with the ``--minL``
             and ``--maxL`` options.
    Step 2 - Ensure that no streamlines end in the cerebrospinal fluid
             according to the provided parcellation. A binary mask can be used
             alternatively through the ``--csf_bin`` option.
    Step 3 - Ensure that no streamlines end in white matter by ensuring that
             they reach the cortical regions according to the provided
             parcellation. The cortical regions of the parcellation can be
             dilated using the ``--ctx_dilation_radius``.
    Step 4 - Remove streamlines if they make a loop with an angle above a
             certain threshold. It's possible to change this angle with the
             ``-a`` option.

Length and loop-based filtering (steps 1 and 2) will not have practical effects
if no specific thresholds are provided (but will be still executed), since
default values are 0 for the minimum allowed length and infinite for the
maximum allowed length and angle.

The anatomical region endings filtering requires a parcellation or label image
file including the cerebrospinal fluid and gray matter (cortical) regions
according to the Desikan-Killiany atlas. Intermediate tractograms (results of
each step and outliers) and volumes can be saved throughout the process.

Example usages:

# Filter length, looping angle and anatomical ending region
>>> scil_filter_tractogram_anatomically.py tractogram.trk wmparc.nii.gz
    path/to/output/directory --minL 20 --maxL 200 -a 300
# Filter only anatomical ending region, with WM dilation and provided csf mask
>>> scil_filter_tractogram_anatomically.py tractogram.trk wmparc.nii.gz
    path/to/output/directory --csf_bin csf_bin.nii.gz --ctx_dilation_radius 2

positional arguments:
  in_tractogram         Path of the input tractogram file.
  in_wmparc             Path of the white matter parcellation atlas
                        (.nii or .nii.gz)
  out_path              Path to the output files.

optional arguments:
  -h, --help            show this help message and exit
  --minL MINL           Minimum length of streamlines, in mm. [0.0]
  --maxL MAXL           Maximum length of streamlines, in mm. [inf]
  -a ANGLE              Maximum looping (or turning) angle of
                        a streamline, in degrees. [inf]
  --csf_bin CSF_BIN     Allow CSF endings filtering with this binary
                        mask instead of using the atlas (.nii or .nii.gz)
  --ctx_dilation_radius CTX_DILATION_RADIUS
                        Cortical labels dilation radius, in mm.
                         [0.0]
  --save_intermediate_tractograms
```

(continues on next page)

(continued from previous page)

```
                        Save accepted and discarded streamlines
                         after each step.
  --save_volumes        Save volumetric images (e.g. binarised label
                         images, etc) in the filtering process.
  --save_counts         Save the streamline counts to a file (.json)
  --no_empty            Do not write file if there is no streamlines.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.

    References:
        [1] Jörgens, D., Descoteaux, M., Moreno, R., 2021. Challenges for
        tractogram ltering. In: Özarslan, E., Schultz, T., Zhang, E., Fuster,
        A. (Eds.), Anisotropy Across Fields and Scales. Springer. Mathematics
        and Visualization.
        [2] Legarreta, J., Petit, L., Rheault, F., Theaud, G., Lemaire, C.,
        Descoteaux, M., Jodoin, P.M. Filtering in tractography using
        autoencoders (FINTA). Medical Image Analysis. 2021
```

2.100 scil_fit_bingham_to_fodf.py

```
usage: __main__.py [-h] [--max_lobes MAX_LOBES] [--at AT] [--rt RT]
                   [--min_sep_angle MIN_SEP_ANGLE]
                   [--max_fit_angle MAX_FIT_ANGLE] [--mask MASK] [-f] [-v]
                   [--processes NBR]
                   in_sh out_bingham

Script for fitting a Bingham distribution to each fODF lobe, as described
in [1].

The Bingham fit is saved, with each Bingham distribution described by 7
coefficients (for example, for a maximum number of lobes of 5, the number
of coefficients is 7 x 5 = 35 -- less than the number of coefficients for
SH of maximum order 8).

Using 12 threads, the execution takes approximately 30 minutes for a brain with
1mm isotropic resolution.

positional arguments:
  in_sh                 Input SH image.
  out_bingham           Output Bingham functions image.

optional arguments:
  -h, --help            show this help message and exit
  --max_lobes MAX_LOBES
                        Maximum number of lobes per voxel to extract. [5]
```

(continues on next page)

(continued from previous page)

```
  --at AT               Absolute threshold for peaks extraction. [0.0]
  --rt RT               Relative threshold for peaks extraction. [0.1]
  --min_sep_angle MIN_SEP_ANGLE
                        Minimum separation angle between two peaks. [25.0]
  --max_fit_angle MAX_FIT_ANGLE
                        Maximum distance in degrees around a peak direction for␣
↪fitting the Bingham function. [15.0]
  --mask MASK           Optional mask file. Only SH inside the mask are fitted.
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]

[1] T. W. Riffert, J. Schreiber, A. Anwander, and T. R. Knösche, "Beyond
    fractional anisotropy: Extraction of bundle-specific structural metrics
    from crossing fiber models," NeuroImage, vol. 100, pp. 176-191, Oct. 2014,
    doi: 10.1016/j.neuroimage.2014.06.015.

[2] J. Schreiber, T. Riffert, A. Anwander, and T. R. Knösche, "Plausibility
    Tracking: A method to evaluate anatomical connectivity and microstructural
    properties along fiber pathways," NeuroImage, vol. 90, pp. 163-178, Apr.
    2014, doi: 10.1016/j.neuroimage.2014.01.002.
```

2.101 scil_fix_dsi_studio_trk.py

```
usage: __main__.py [-h] [--in_native_fa IN_NATIVE_FA] [--auto_crop]
                   [--save_transfo FILE | --load_transfo FILE]
                   [--cut_invalid | --remove_invalid] [-f] [--no_bbox_check]
                   in_dsi_tractogram in_dsi_fa out_tractogram

This script is made to fix DSI-Studio TRK file (unknown space/convention) to
make it compatible with TrackVis, MI-Brain, Dipy Horizon (Stateful Tractogram).

The script either make it match with an anatomy from DSI-Studio (AC-PC aligned,
sometimes flipped) or if --in_native_fa is provided it moves it back to native
DWI space (this involved registration).

Since DSI-Studio sometimes leaves some skull around the brain, the --auto_crop
aims to stabilize registration. If this option fails, manually BET both FA.
Registration is more robust at resolution above 2mm (iso), be careful.

If you are fixing bundles, use this script once with --save_transfo and verify
results. Once satisfied, call the scripts on bundles using a bash for loop with
--load_transfo to save computation.

We recommand the --cut_invalid to remove invalid points of streamlines rather
removing entire streamlines.

This script was tested on various datasets and worked on all of them. However,
always verify the results and if a specific case does not work. Open an issue
on the Scilpy GitHub repository.

WARNING: This script is still experimental, DSI-Studio evolves quickly and
results may vary depending on the data itself as well as DSI-studio version.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_dsi_tractogram     Path of the input tractogram file from DSI studio (.trk).
  in_dsi_fa             Path of the input FA from DSI Studio (.nii.gz).
  out_tractogram        Path of the output tractogram file.

optional arguments:
  -h, --help            show this help message and exit
  --in_native_fa IN_NATIVE_FA
                        Path of the input FA from Dipy/MRtrix (.nii.gz).
                        Move the tractogram back to a "proper" space,␣
↪includeregistration.
  --auto_crop           If both FA are not already BET, perform registration
                        using a centered-cube crop to ignore the skull.
                        A good BET for both is more robust.
  --save_transfo FILE   Save estimated transformation to avoid recomputing (.txt).
  --load_transfo FILE   Load estimated transformation to apply to other files (.txt).
  --cut_invalid         Cut invalid streamlines rather than removing them.
                        Keep the longest segment only.
  --remove_invalid      Remove the streamlines landing out of the bounding box.
  -f                    Force overwriting of the output files.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).
```

2.102 scil_flip_gradients.py

```
usage: __main__.py [-h] (--fsl | --mrtrix) [-f]
                   gradient_sampling_file flipped_sampling_file dimension
                   [dimension ...]

Flip one or more axes of the gradient sampling matrix. It will be saved in
the same format as input gradient sampling file.

positional arguments:
  gradient_sampling_file
                        Path to gradient sampling file. (.bvec or .b)
  flipped_sampling_file
                        Path to the flipped gradient sampling file.
  dimension             The axes you want to flip. eg: to flip the x and y axes use:␣
↪x y.

optional arguments:
  -h, --help            show this help message and exit
  --fsl                 Specify fsl format.
  --mrtrix              Specify mrtrix format.
  -f                    Force overwriting of the output files.
```

2.103 scil_flip_streamlines.py

```
usage: __main__.py [-h] [--reference REFERENCE] [-f]
                   in_tractogram out_tractogram {x,y,z} [{x,y,z} ...]

Flip streamlines locally around specific axes.

IMPORTANT: this script should only be used in case of absolute necessity.
It's better to fix the real tools than to force flipping streamlines to
have them fit in the tools.

positional arguments:
  in_tractogram         Path of the input tractogram file.
  out_tractogram        Path of the output tractogram file.
  {x,y,z}               The axes you want to flip. eg: to flip the x and y axes use:␣
↪x y.

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.104 scil_flip_surface.py

```
usage: __main__.py [-h] [-f] in_surface out_surface {x,y,z,n} [{x,y,z,n} ...]

Script to flip and reverse a surface (FreeSurfer or VTK supported).
Can be used to flip in chosen axes (x, y or z),
it can also flip inside out the surface orientation (normal).

Best usage for FreeSurfer to LPS vtk (for MI-Brain):
!!! important FreeSurfer surfaces must be in their respective folder !!!
> mris_convert --to-scanner lh.white lh.white.vtk
> scil_flip_surface.py lh.white.vtk lh_white_lps.vtk x y

positional arguments:
  in_surface    Input surface (.vtk).
  out_surface   Output flipped surface (.vtk).
  {x,y,z,n}     The axes (or normal orientation) you want to flip. eg: to flip the x␣
↪and y axes use: x y.

optional arguments:
  -h, --help    show this help message and exit
  -f            Force overwriting of the output files.

References:
[1] St-Onge, E., Daducci, A., Girard, G. and Descoteaux, M. 2018.
    Surface-enhanced tractography (SET). NeuroImage.
```

2.105 scil_flip_volume.py

```
usage: __main__.py [-h] [-f] in_image out_image dimension [dimension ...]

Flip the volume according to the specified axis.

positional arguments:
  in_image    Path of the input volume (nifti).
  out_image   Path of the output volume (nifti).
  dimension   The axes you want to flip. eg: to flip the x and y axes use: x y.

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
```

2.106 scil_generate_gradient_sampling.py

```
usage: __main__.py [-h] [--eddy] [--duty] [--b0_every B0_EVERY] [--b0_end]
                   [--b0_value B0_VALUE] [--b0_philips]
                   (--bvals bvals [bvals ...] | --b_lin_max B_LIN_MAX | --q_lin_max Q_
↪LIN_MAX)
                   [--fsl] [--mrtrix] [-v] [-f]
                   nb_samples [nb_samples ...] out_basename

Generate multi-shell gradient sampling with various processing to accelerate
acquisition and help artefact correction.

Multi-shell gradient sampling is generated as in [1], the bvecs are then
flipped to maximize spread for eddy current correction, b0s are interleaved
at equal spacing and the non-b0 samples are finally shuffled
to minimize the total diffusion gradient amplitude over a few TR.

positional arguments:
  nb_samples            Number of samples on each non b0 shell. If multishell,
                        provide a number per shell.
  out_basename          Gradient sampling output basename (don't include
                        extension). Please add options --fsl and/or --mrtrix
                        below.

Options and Parameters:
  -h, --help            show this help message and exit
  --eddy                Apply eddy optimization. B-vectors are flipped to be
                        well spread without symmetry. [False]
  --duty                Apply duty cycle optimization. B-vectors are shuffled
                        to reduce consecutive colinearity in the samples.
                        [False]
  --b0_every B0_EVERY   Interleave a b0 every b0_every. No b0 if 0. Only 1 b0
                        at beginning if > number of samples or negative. [-1]
  --b0_end              Add a b0 as last sample. [False]
  --b0_value B0_VALUE   b-value of the b0s. [0.0]
  --b0_philips          Replace values of b0s bvecs by existing bvecs for
                        Philips handling. [False]
  --bvals bvals [bvals ...]
                        bval of each non-b0 shell.
```

(continues on next page)

(continued from previous page)

```
  --b_lin_max B_LIN_MAX
                        b-max for linear bval distribution in *b*. [replaces
                        -bvals]
  --q_lin_max Q_LIN_MAX
                        b-max for linear bval distribution in *q*. [replaces
                        -bvals]
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Save as:
  --fsl                 Save in FSL format (.bvec/.bval). [False]
  --mrtrix              Save in MRtrix format (.b). [False]

References: [1] Emmanuel Caruyer, Christophe Lenglet, Guillermo Sapiro,
Rachid Deriche. Design of multishell gradient sampling with uniform coverage
in diffusion MRI. Magnetic Resonance in Medicine, Wiley, 2013, 69 (6),
pp. 1534-1540. <http://dx.doi.org/10.1002/mrm.24736>
```

2.107 scil_generate_priors_from_bundle.py

```
usage: __main__.py [-h] [--sh_basis {descoteaux07,tournier07}]
                   [--todi_sigma {0,1,2,3,4}] [--sf_threshold SF_THRESHOLD]
                   [--out_prefix OUT_PREFIX] [--out_dir OUT_DIR] [-f]
                   [--reference REFERENCE]
                   in_bundle in_fodf in_mask

Generation of priors and enhanced-FOD from an example/template bundle.
The bundle must have been cleaned thorougly before use. The E-FOD can then
be used for bundle-specific tractography, but not for FOD metrics.

positional arguments:
  in_bundle             Input bundle filename.
  in_fodf               Input FOD filename.
  in_mask               Mask to constrain the TODI spatial smoothing,
                        for example a WM mask.

optional arguments:
  -h, --help            show this help message and exit
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --todi_sigma {0,1,2,3,4}
                        Smooth the orientation histogram.
  --sf_threshold SF_THRESHOLD
                        Relative threshold for sf masking (0.0-1.0).
  --out_prefix OUT_PREFIX
                        Add a prefix to all output filename,
                        default is no prefix.
  --out_dir OUT_DIR     Output directory for all generated files,
                        default is current directory.
```

(continues on next page)

(continued from previous page)

```
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).

    References:
        [1] Rheault, Francois, et al. "Bundle-specific tractography with
        incorporated anatomical and orientational priors."
        NeuroImage 186 (2019): 382-398
```

2.108 scil_group_comparison.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR] [--out_json OUT_JSON]
                   [--bundles BUNDLES [BUNDLES ...]]
                   [--metrics METRICS [METRICS ...]]
                   [--values VALUES [VALUES ...]] [--alpha_error ALPHA_ERROR]
                   [--generate_graph] [--indent INDENT] [--sort_keys] [-v]
                   [-f]
                   IN_JSON IN_PARTICIPANTS GROUP_BY

Run group comparison statistics on metrics from tractometry
1) Separate the sample given a particular variable (group_by) into groups

2) Does Shapiro-Wilk test of normality for every sample
https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test

3) Does Levene or Bartlett (depending on normality) test of variance
homogeneity Levene:
https://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm
Bartlett:
https://www.itl.nist.gov/div898/handbook/eda/section3/eda357.htm

4) Test the group difference for every measure with the correct test depending
   on the sample (Student, Welch, Mannwhitneyu, ANOVA, Kruskall-Wallis)
Student :
https://en.wikipedia.org/wiki/Student%27s_t-test#Independent_two-sample_t-test
Welch :
https://en.wikipedia.org/wiki/Welch%27s_t-test
Mann-Whitney U :
https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test
ANOVA :
http://www.biostathandbook.com/onewayanova.html
Kruskall-Wallis :
https://en.wikipedia.org/wiki/Kruskal%E2%80%93Wallis_one-way_analysis_of_variance

5) If the group difference test is positive and number of group is greater than
   2, test the group difference two by two.

6) Generate the result for all metrics and bundles

positional arguments:
  IN_JSON               Input JSON file from tractometry nextflow pipeline or␣
↪equivalent.
  IN_PARTICIPANTS       Input tsv participants file.See doc in https://scilpy.
↪readthedocs.io/en/latest/documentation/construct_participants_tsv_file.html
```

(continues on next page)

(continued from previous page)

```
  GROUP_BY              Variable that will be used to compare group together.

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Name of the output folder path. [stats]
  --out_json OUT_JSON   The name of the result json output file otherwise it will be␣
↪printed.
  --bundles BUNDLES [BUNDLES ...], -b BUNDLES [BUNDLES ...]
                        Bundle(s) in which you want to do stats. [all]
  --metrics METRICS [METRICS ...], -m METRICS [METRICS ...]
                        Metric(s) on which you want to do stats. [all]
  --values VALUES [VALUES ...], --va VALUES [VALUES ...]
                        Value(s) on which you want to do stats (mean, std). [all]
  --alpha_error ALPHA_ERROR, -a ALPHA_ERROR
                        Type 1 error for all the test. [0.05]
  --generate_graph, --gg
                        Generate a simple plot of every metric across groups.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.109 scil_image_math.py

```
usage: __main__.py [-h] [--data_type DATA_TYPE] [--exclude_background] [-f]
                   [-v]
                   {lower_threshold,upper_threshold,lower_threshold_eq,upper_
↪threshold_eq,lower_clip,upper_clip,absolute_value,round,ceil,floor,normalize_sum,
↪normalize_max,log_10,log_e,convert,invert,addition,subtraction,multiplication,
↪division,mean,std,union,intersection,difference,concatenate,dilation,erosion,
↪closing,opening,blur}
                   in_images [in_images ...] out_image

Performs an operation on a list of images. The supported operations are
listed below.

This script is loading all images in memory, will often crash after a few
hundred images.

Some operations such as multiplication or addition accept float value as
parameters instead of images.
> scil_image_math.py multiplication img.nii.gz 10 mult_10.nii.gz

    lower_threshold: IMG THRESHOLD
        All values below the threshold will be set to zero.
        All values above the threshold will be set to one.

    upper_threshold: IMG THRESHOLD
        All values below the threshold will be set to one.
        All values above the threshold will be set to zero.
        Equivalent to lower_threshold followed by an inversion.
```

(continues on next page)

(continued from previous page)

```
    lower_threshold_eq: IMG THRESHOLD
        All values below the threshold will be set to zero.
        All values above or equal the threshold will be set to one.

    upper_threshold_eq: IMG THRESHOLD
        All values below or equal the threshold will be set to one.
        All values above the threshold will be set to zero.
        Equivalent to lower_threshold followed by an inversion.

    lower_clip: IMG THRESHOLD
        All values below the threshold will be set to threshold.

    upper_clip: IMG THRESHOLD
        All values above the threshold will be set to threshold.

    absolute_value: IMG
        All negative values will become positive.

    round: IMG
        Round all decimal values to the closest integer.

    ceil: IMG
        Ceil all decimal values to the next integer.

    floor: IMG
        Floor all decimal values to the previous integer.

    normalize_sum: IMG
        Normalize the image so the sum of all values is one.

    normalize_max: IMG
        Normalize the image so the maximum value is one.

    log_10: IMG
        Apply a log (base 10) to all non zeros values of an image.

    log_e: IMG
        Apply a natural log to all non zeros values of an image.

    convert: IMG
        Perform no operation, but simply change the data type.

    invert: IMG
        Operation on binary image to interchange 0s and 1s in a binary mask.

    addition: IMGs
        Add multiple images together.

    subtraction: IMG_1 IMG_2
        Subtract first image by the second (IMG_1 - IMG_2).

    multiplication: IMGs
        Multiply multiple images together (danger of underflow and overflow)

    division: IMG_1 IMG_2
        Divide first image by the second (danger of underflow and overflow)
        Ignore zeros values, excluded from the operation.
```

(continues on next page)

(continued from previous page)

```
    mean: IMGs
        Compute the mean of images.
        If a single 4D image is provided, average along the last dimension.

    std: IMGs
        Compute the standard deviation average of multiple images.
        If a single 4D image is provided, compute the STD along the last
        dimension.

    union: IMGs
        Operation on binary image to keep voxels, that are non-zero, in at
        least one file.

    intersection: IMGs
        Operation on binary image to keep the voxels, that are non-zero,
        are present in all files.

    difference: IMG_1 IMG_2
        Operation on binary image to keep voxels from the first file that are
        not in the second file (non-zeros).

    concatenate: IMGs
        Concatenate a list of 3D and 4D images into a single 4D image.

    dilation: IMG, VALUE
        Binary morphological operation to spatially extend the values of an
        image to their neighbors. VALUE is in voxels.

    erosion: IMG, VALUE
        Binary morphological operation to spatially shrink the volume contained
        in a binary image. VALUE is in voxels.

    closing: IMG, VALUE
        Binary morphological operation, dilation followed by an erosion.

    opening: IMG, VALUE
        Binary morphological operation, erosion followed by a dilation.

    blur: IMG, VALUE
        Apply a gaussian blur to a single image.


positional arguments:
  {lower_threshold,upper_threshold,lower_threshold_eq,upper_threshold_eq,lower_clip,
↪upper_clip,absolute_value,round,ceil,floor,normalize_sum,normalize_max,log_10,log_e,
↪convert,invert,addition,subtraction,multiplication,division,mean,std,union,
↪intersection,difference,concatenate,dilation,erosion,closing,opening,blur}
                        The type of operation to be performed on the images.
  in_images             The list of image files or parameters.
  out_image             Output image path.

optional arguments:
  -h, --help            show this help message and exit
  --data_type DATA_TYPE
                        Data type of the output image. Use the format: uint8, int16,␣
↪int/float32, int/float64.
```

(continues on next page)

(continued from previous page)

```
  --exclude_background  Does not affect the background of the original images.
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.110 scil_merge_json.py

```
usage: __main__.py [-h] [--keep_separate] [--no_list]
                   [--add_parent_key ADD_PARENT_KEY] [--remove_parent_key]
                   [--recursive] [--average_last_layer] [--indent INDENT]
                   [--sort_keys] [-f]
                   in_json [in_json ...] out_json

 Merge multiple json file into a single one.
the --keep_separate option will add an entry for each file, the basename will
become the key.

positional arguments:
  in_json               List of json files to merge (.json).
  out_json              Output json file (.json).

optional arguments:
  -h, --help            show this help message and exit
  --keep_separate       Merge entries as separate keys based on filename.
  --no_list             Merge entries knowing there is no conflict.
  --add_parent_key ADD_PARENT_KEY
                        Merge all entries under a single parent.
  --remove_parent_key   Merge ignoring parent key (e.g for population).
  --recursive           Merge all entries at the lowest layers.
  --average_last_layer  Average all entries at the lowest layers.
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.111 scil_merge_sh.py

```
usage: __main__.py [-h] [-f] in_shs [in_shs ...] out_sh

Merge a list of Spherical Harmonics files.

This merges the coefficients of multiple Spherical Harmonics files
by taking, for each coefficient, the one with the largest magnitude.

Can be used to merge fODFs computed from different shells into 1, while
conserving the most relevant information.

Based on [1].

positional arguments:
  in_shs      List of SH files.
```

(continues on next page)

(continued from previous page)

```
  out_sh      output SH file.

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.

Reference:
[1] Garyfallidis, E., Zucchelli, M., Houde, J-C., Descoteaux, M.
    How to perform best ODF reconstruction from the Human Connectome
    Project sampling scheme?
    ISMRM 2014.
```

2.112 scil_normalize_connectivity.py

```
usage: __main__.py [-h]
                   [--length LENGTH_MATRIX | --inverse_length LENGTH_MATRIX]
                   [--bundle_volume VOLUME_MATRIX]
                   [--parcel_volume ATLAS LABELS_LIST | --parcel_surface ATLAS LABELS_
↪LIST]
                   [--max_at_one | --sum_to_one | --log_10] [-f]
                   in_matrix out_matrix

Normalize a connectivity matrix coming from scil_decompose_connectivity.py.
3 categories of normalization are available:
-- Edge attributes
 - length: Multiply each edge by the average bundle length.
   Compensate for far away connections when using interface seeding.
   Cannot be used with inverse_length.

 - inverse_length: Divide each edge by the average bundle length.
   Compensate for big connections when using white matter seeding.
   Cannot be used with length.

 - bundle_volume: Divide each edge by the average bundle length.
   Compensate for big connections when using white matter seeding.

-- Node attributes (Mutually exclusive)
 - parcel_volume: Divide each edge by the sum of node volume.
   Compensate for the likelihood of ending in the node.
   Compensate seeding bias when using interface seeding.

 - parcel_surface: Divide each edge by the sum of the node surface.
   Compensate for the likelihood of ending in the node.
   Compensate for seeding bias when using interface seeding.

-- Matrix scaling (Mutually exclusive)
 - max_at_one: Maximum value of the matrix will be set to one.
 - sum_to_one: Ensure the sum of all edges weight is one
 - log_10: Apply a base 10 logarithm to all edges weight

The volume and length matrix should come from the
scil_decompose_connectivity.py script.

A review of the type of normalization is available in:
```

(continues on next page)

(continued from previous page)

```
Colon-Perez, Luis M., et al. "Dimensionless, scale-invariant, edge weight
metric for the study of complex structural networks." PLOS one 10.7 (2015).

However, the proposed weighting of edge presented in this publication is not
implemented.

positional arguments:
  in_matrix             Input connectivity matrix. This is typically a streamline_
↪count matrix (.npy).
  out_matrix            Output normalized matrix (.npy).

optional arguments:
  -h, --help            show this help message and exit
  -f                    Force overwriting of the output files.

Edge-wise options:
  --length LENGTH_MATRIX
                        Length matrix used for edge-wise multiplication.
  --inverse_length LENGTH_MATRIX
                        Length matrix used for edge-wise division.
  --bundle_volume VOLUME_MATRIX
                        Volume matrix used for edge-wise division.
  --parcel_volume ATLAS LABELS_LIST
                        Atlas and labels list for edge-wise division.
  --parcel_surface ATLAS LABELS_LIST
                        Atlas and labels list for edge-wise division.

Scaling options:
  --max_at_one          Scale matrix with maximum value at one.
  --sum_to_one          Scale matrix with sum of all elements at one.
  --log_10              Apply a base 10 logarithm to the matrix.
```

2.113 scil_outlier_rejection.py

```
usage: __main__.py [-h] [--remaining_bundle REMAINING_BUNDLE] [--alpha ALPHA]
                   [--display_counts] [--reference REFERENCE] [-f]
                   [--indent INDENT] [--sort_keys]
                   in_bundle out_bundle

Clean a bundle (inliers/outliers) using hiearchical clustering.
http://archive.ismrm.org/2015/2844.html

If spurious streamlines are dense, it is possible they will not be recognized
as outliers. Manual cleaning may be required to overcome this limitation.

positional arguments:
  in_bundle             Fiber bundle file to remove outliers from.
  out_bundle            Fiber bundle without outliers.

optional arguments:
  -h, --help            show this help message and exit
  --remaining_bundle REMAINING_BUNDLE
                        Removed outliers.
  --alpha ALPHA         Percent of the length of the tree that clusters of individual␣
↪streamlines will be pruned. [0.6]
```

(continues on next page)

(continued from previous page)

```
  --display_counts      Print streamline count before and after filtering
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.114 scil_perform_majority_vote.py

```
usage: __main__.py [-h] [--ratio_streamlines RATIO_STREAMLINES]
                   [--ratio_voxels RATIO_VOXELS] [--same_tractogram]
                   [--output_prefix OUTPUT_PREFIX] [--reference REFERENCE]
                   [-f]
                   in_bundles [in_bundles ...]

Use multiple bundles to perform a voxel-wise vote (occurence across input).
If streamlines originate from the same tractogram, streamline-wise vote
is available.

Useful to generate an average representation from bundles of a given population
or multiple bundle segmentations (gold standard).

Input tractograms must have identical header.

positional arguments:
  in_bundles            Input bundles filename.

optional arguments:
  -h, --help            show this help message and exit
  --ratio_streamlines RATIO_STREAMLINES
                        Minimum vote to be considered for streamlines [0.5].
  --ratio_voxels RATIO_VOXELS
                        Minimum vote to be considered for voxels [0.5].
  --same_tractogram     All bundles need to come from the same tractogram,
                        will generate a voting for streamlines too.
  --output_prefix OUTPUT_PREFIX
                        Output prefix, [voting_].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.115 scil_plot_mean_std_per_point.py

```
usage: __main__.py [-h] [--stats_over_population] [--nb_pts NB_PTS]
                   [--display_means]
                   [--fill_color FILL_COLOR | --dict_colors DICT_COLORS] [-f]
                   in_json out_dir
```

(continues on next page)

(continued from previous page)

```
Plot all mean/std per point for a subject or population json file from
tractometry-flow.
WARNING: For population, the displayed STDs is only showing the variation
of the means. It does not account intra-subject STDs.

positional arguments:
  in_json               JSON file containing the mean/std per point. For example, can␣
↪be created using scil_compute_metrics_along_streamline.
  out_dir               Output directory.

optional arguments:
  -h, --help            show this help message and exit
  --stats_over_population
                        If set, consider the input stats to be over an entire␣
↪population and not subject-based.
  --nb_pts NB_PTS       Force the number of divisions for the bundles.
                        Avoid unequal plots across datasets, replace missing data␣
↪with zeros.
  --display_means       Display the subjects means as semi-transparent line.
                        Poor results when the number of subject is high.
  --fill_color FILL_COLOR
                        Hexadecimal RGB color filling the region between mean +/- std.
↪ The hexadecimal RGB color should be formatted as 0xRRGGBB.
  --dict_colors DICT_COLORS
                        Dictionnary mapping basename to color.Same convention as --
↪color.
  -f                    Force overwriting of the output files.
```

2.116 scil_prepare_eddy_command.py

```
usage: __main__.py [-h] [--n_reverse N_REVERSE] [--topup TOPUP]
                   [--topup_params TOPUP_PARAMS]
                   [--eddy_cmd {eddy_openmp,eddy_cuda,eddy_cuda8.0,eddy_cuda9.1,eddy_
↪cuda10.2,eddy,eddy_cpu}]
                   [--b0_thr B0_THR] [--encoding_direction {x,y,z}]
                   [--readout READOUT] [--slice_drop_correction]
                   [--lsr_resampling] [--out_directory OUT_DIRECTORY]
                   [--out_prefix OUT_PREFIX] [--out_script] [--fix_seed]
                   [--eddy_options EDDY_OPTIONS] [-f] [-v]
                   in_dwi in_bvals in_bvecs in_mask

Prepare a typical command for eddy and create the necessary files. When using
multiple acquisitions and/or opposite phase directions, images, b-values and
b-vectors should be merged together using scil_concatenate_dwi.py. If using
topup prior to calling this script, images should be concatenated in the same
order as the b0s used with prepare_topup.

positional arguments:
  in_dwi                Input DWI Nifti image. If using multiple acquisition and/or␣
↪opposite phase directions, please merge in the same order as for prepare_topup␣
↪using scil_concatenate_dwi.py.
  in_bvals              Input b-values file in FSL format.
  in_bvecs              Input b-vectors file in FSL format.
```

(continues on next page)

(continued from previous page)

```
  in_mask               Binary brain mask.

optional arguments:
  -h, --help            show this help message and exit
  --n_reverse N_REVERSE
                        Number of reverse phase volumes included in the DWI image [0].
  --topup TOPUP         Topup output name. If given, apply topup during eddy.
                        Should be the same as --out_prefix from scil_prepare_topup_
↪command.py.
  --topup_params TOPUP_PARAMS
                        Parameters file (typically named acqparams) used to run topup.
  --eddy_cmd {eddy_openmp,eddy_cuda,eddy_cuda8.0,eddy_cuda9.1,eddy_cuda10.2,eddy,eddy_
↪cpu}
                        Eddy command [eddy_openmp].
  --b0_thr B0_THR       All b-values with values less than or equal to b0_thr are␣
↪considered
                        as b0s i.e. without diffusion weighting [20].
  --encoding_direction {x,y,z}
                        Acquisition direction, default is AP-PA [y].
  --readout READOUT     Total readout time from the DICOM metadata [0.062].
  --slice_drop_correction
                        If set, will activate eddy's outlier correction,
                        which includes slice drop correction.
  --lsr_resampling      Perform least-square resampling, allowing eddy to combine␣
↪forward and reverse phase acquisitions for better reconstruction. Only works if␣
↪directions and b-values are identical in both phase direction.
  --out_directory OUT_DIRECTORY
                        Output directory for eddy files [.].
  --out_prefix OUT_PREFIX
                        Prefix of the eddy-corrected DWI [dwi_eddy_corrected].
  --out_script          If set, will output a .sh script (eddy.sh).
                        else, will output the lines to the terminal [False].
  --fix_seed            If set, will use the fixed seed strategy for eddy.
                        Enhances reproducibility.
  --eddy_options EDDY_OPTIONS
                        Additional options you want to use to run eddy.
                        Add these options using quotes (i.e. "--ol_nstd=6 --mb=4").
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.117 scil_prepare_topup_command.py

```
usage: __main__.py [-h] [--config CONFIG] [--encoding_direction {x,y,z}]
                   [--readout READOUT] [--out_b0s OUT_B0S]
                   [--out_directory OUT_DIRECTORY] [--out_prefix OUT_PREFIX]
                   [--out_params OUT_PARAMS] [--out_script]
                   [--topup_options TOPUP_OPTIONS] [-f] [-v]
                   in_forward_b0 in_reverse_b0

Prepare a typical command for topup and create the necessary files.
The reversed b0 must be in a different file.

positional arguments:
  in_forward_b0         Input b0 Nifti image with forward phase encoding.
```

(continues on next page)

(continued from previous page)

```
  in_reverse_b0         Input b0 Nifti image with reversed phase encoding.

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Topup config file [b02b0.cnf].
  --encoding_direction {x,y,z}
                        Acquisition direction of the forward b0 image, default is AP␣
↪[y].
  --readout READOUT     Total readout time from the DICOM metadata [0.062].
  --out_b0s OUT_B0S     Output fused b0 file [fused_b0s.nii.gz].
  --out_directory OUT_DIRECTORY
                        Output directory for topup files [.].
  --out_prefix OUT_PREFIX
                        Prefix of the topup results [topup_results].
  --out_params OUT_PARAMS
                        Filename for the acquisition parameters file [acqparams.txt].
  --out_script          If set, will output a .sh script (topup.sh).
                        else, will output the lines to the terminal [False].
  --topup_options TOPUP_OPTIONS
                        Additional options you want to use to run topup.
                        Add these options using quotes (i.e. "--fwhm=6 --miter=4").
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.118 scil_print_connectivity_filenames.py

```
usage: __main__.py [-h] [-f] in_matrix labels_list out_txt

Output the list of filenames using the coordinates from a binary connectivity
matrix. Typically used to move around files that are considered valid after
the scil_filter_connectivity.py script.

Example:
# Keep connections with more than 1000 streamlines for 100% of a population
scil_filter_connectivity.py filtering_mask.npy
    --greater_than */streamlines_count.npy 1000 1.0
scil_print_connectivity_filenames.py filtering_mask.npy
    labels_list.txt pass.txt
for file in $(cat pass.txt);
    do mv ${SOMEWHERE}/${FILE} ${SOMEWHERE_ELSE}/;
done

positional arguments:
  in_matrix     Binary matrix in numpy (.npy) format.
                Typically from scil_filter_connectivity.py
  labels_list   List saved by the decomposition script.
  out_txt       Output text file containing all filenames.

optional arguments:
  -h, --help    show this help message and exit
  -f            Force overwriting of the output files.
```

2.119 scil_print_header.py

```
usage: __main__.py [-h] [--keys KEYS [KEYS ...]] in_file

Print the raw header from the provided file or only the specified keys.
Support trk, nii and mgz files.

positional arguments:
  in_file               Input file (trk, nii and mgz).

optional arguments:
  -h, --help            show this help message and exit
  --keys KEYS [KEYS ...]
                        Print only the specified keys.
```

2.120 scil_project_streamlines_to_map.py

```
usage: __main__.py [-h]
                   (--in_metrics IN_METRICS [IN_METRICS ...] | --use_dps DPS_KEY [DPS_
↪KEY ...] | --use_dpp DPP_KEY [DPP_KEY ...] | --load_dps DPS_KEY [DPS_KEY ...] | --
↪load_dpp DPP_KEY [DPP_KEY ...])
                   [--from_wm] [--to_wm] [--reference REFERENCE] [-f]
                   in_bundle out_folder

Projects metrics onto the endpoints of streamlines. The idea is to visualize
the cortical areas affected by metrics (assuming streamlines start/end in
the cortex).

This script can project data from maps (--in_metrics), from data_per_point
(dpp) or data_per_streamline (dps): --load_dpp and --load_dps require an array
from a file (must be the right shape), --use_dpp and --use_dps work only for
.trk file and the key must exist in the metadata.

The default options will take data from endpoints and project it to endpoints.
--from_wm will use data from whole streamlines.
--to_wm will project the data to whole streamline coverage.
This creates 4 combinations of data source and projection.

positional arguments:
  in_bundle             Fiber bundle file.
  out_folder            Folder where to save endpoints metric.

optional arguments:
  -h, --help            show this help message and exit
  --in_metrics IN_METRICS [IN_METRICS ...]
                        Nifti metric(s) to compute statistics on.
  --use_dps DPS_KEY [DPS_KEY ...]
                        Use the data_per_streamline (scalar) from file, e.g. commit_
↪weights.
  --use_dpp DPP_KEY [DPP_KEY ...]
                        Use the data_per_point (scalar) from file.
  --load_dps DPS_KEY [DPS_KEY ...]
                        Load data per streamline (scalar) .txt or .npy.
  --load_dpp DPP_KEY [DPP_KEY ...]
```

(continues on next page)

(continued from previous page)

```
                        Load data per point (scalar) from .txt or .npy.
  --from_wm             Project metrics from whole streamlines coverage.
  --to_wm               Project metrics into streamlines coverage.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.121 scil_recognize_multi_bundles.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR]
                   [--log_level {DEBUG,INFO,WARNING,ERROR}]
                   [--multi_parameters MULTI_PARAMETERS]
                   [--minimal_vote_ratio MINIMAL_VOTE_RATIO]
                   [--tractogram_clustering_thr TRACTOGRAM_CLUSTERING_THR [TRACTOGRAM_
↪CLUSTERING_THR ...]]
                   [--seeds SEEDS [SEEDS ...]] [--inverse] [--processes NBR]
                   [-f]
                   in_tractogram in_config_file in_models_directories
                   [in_models_directories ...] in_transfo

Compute RecobundlesX (multi-atlas & multi-parameters).
The model needs to be cleaned and lightweight.
Transform should come from ANTs: (using the --inverse flag)
AntsRegistrationSyNQuick.sh -d 3 -m MODEL_REF -f SUBJ_REF

If you are not sure about the transformation 'direction' you can try
scil_recognize_single_bundle.py (with the -v option), a warning will popup if
the provided transformation is not use correctly.

The next two arguments are multi-parameters related:
--multi_parameters must be lower than len(model_clustering_thr) *
len(bundle_pruning_thr) * len(tractogram_clustering_thr)

--seeds can be more than one value. Multiple values will result in
a overall multiplicative factor of len(seeds) * '--multi_parameters'

The number of folders provided by 'models_directories' will further multiply
the total number of runs. Meaning that the total number of Recobundles
execution will be len(seeds) * '--multi_parameters' * len(models_directories)

--minimal_vote_ratio is a value between 0 and 1. The actual number of votes
required will be '--minimal_vote_ratio' * len(seeds) * '--multi_parameters'
* len(models_directories).

Example: 5 atlas, 9 multi-parameters, 2 seeds with a minimal vote_ratio
of 0.50 will results in 90 executions (for each bundle in the config file)
and a minimal vote of 45 / 90.

Example data and usage available at: https://zenodo.org/record/3928503

positional arguments:
  in_tractogram         Input tractogram filename (.trk or .tck).
  in_config_file        Path of the config file (.json)
```

(continues on next page)

(continued from previous page)

```
  in_models_directories
                        Path for the directories containing model.
  in_transfo            Path for the transformation to model space (.txt, .npy or .
↪mat).

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Path for the output directory [voting_results].
  --log_level {DEBUG,INFO,WARNING,ERROR}
                        Log level of the logging class.
  --multi_parameters MULTI_PARAMETERS
                        Pick parameters from the potential combinations
                        Will multiply the number of times Recobundles is ran.
                        See the documentation [1].
  --minimal_vote_ratio MINIMAL_VOTE_RATIO
                        Streamlines will only be considered for saving if
                        recognized often enough [0.5].
  --tractogram_clustering_thr TRACTOGRAM_CLUSTERING_THR [TRACTOGRAM_CLUSTERING_THR ...
↪]
                        Input tractogram clustering thresholds [12]mm.
  --seeds SEEDS [SEEDS ...]
                        Random number generator seed [0]
                        Will multiply the number of times Recobundles is ran.
  --inverse             Use the inverse transformation.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.

Garyfallidis, E., Cote, M. A., Rheault, F., ... &
Descoteaux, M. (2018). Recognition of white matter
bundles using local and global streamline-based registration and
clustering. NeuroImage, 170, 283-295.
```

2.122 scil_recognize_single_bundle.py

```
usage: __main__.py [-h]
                   [--tractogram_clustering_thr TRACTOGRAM_CLUSTERING_THR]
                   [--model_clustering_thr MODEL_CLUSTERING_THR]
                   [--pruning_thr PRUNING_THR] [--slr_threads SLR_THREADS]
                   [--seed SEED] [--inverse] [--no_empty]
                   [--in_pickle IN_PICKLE | --out_pickle OUT_PICKLE]
                   [--reference REFERENCE] [-v] [-f]
                   in_tractogram in_model in_transfo out_tractogram

Compute a simple Recobundles (single-atlas & single-parameters).
The model need to be cleaned and lightweight.
Transform should come from ANTs: (using the --inverse flag)
AntsRegistrationSyNQuick.sh -d 3 -m MODEL_REF -f SUBJ_REF

If you are unsure about the transformation 'direction' use the verbose
option (-v) and try with and without the --inverse flag. If you are not using
the right transformation 'direction' a warning will popup. If there is no
warning in both case it means the transformation is very close to identity and
both 'direction' will work.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  in_tractogram         Input tractogram filename.
  in_model              Model to use for recognition.
  in_transfo            Path for the transformation to model space (.txt, .npy or .
↪mat).
  out_tractogram        Output tractogram filename.

optional arguments:
  -h, --help            show this help message and exit
  --tractogram_clustering_thr TRACTOGRAM_CLUSTERING_THR
                        Clustering threshold used for the whole brain [8mm].
  --model_clustering_thr MODEL_CLUSTERING_THR
                        Clustering threshold used for the model [4mm].
  --pruning_thr PRUNING_THR
                        MDF threshold used for final streamlines selection [6mm].
  --slr_threads SLR_THREADS
                        Number of threads for SLR [1].
  --seed SEED           Random number generator seed [None].
  --inverse             Use the inverse transformation.
  --no_empty            Do not write file if there is no streamline.
  --in_pickle IN_PICKLE
                        Input pickle clusters map file.
                        Will override the tractogram_clustering_thr parameter.
  --out_pickle OUT_PICKLE
                        Output pickle clusters map file.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

Garyfallidis, E., Cote, M. A., Rheault, F., ... &
Descoteaux, M. (2018). Recognition of white matter
bundles using local and global streamline-based registration and
clustering. NeuroImage, 170, 283-295.
```

2.123 scil_register_tractogram.py

```
usage: __main__.py [-h] [--out_name OUT_NAME] [--only_rigid]
                   [--moving_tractogram_ref MOVING_TRACTOGRAM_REF]
                   [--static_tractogram_ref STATIC_TRACTOGRAM_REF] [-f] [-v]
                   moving_tractogram static_tractogram

Generate a linear transformation matrix from the registration of
2 tractograms. Typically, this script is run before
scil_apply_transform_to_tractogram.py.

For more informations on how to use the various registration scripts
see the doc/tractogram_registration.md readme file

positional arguments:
  moving_tractogram     Path of the moving tractogram.
  static_tractogram     Path of the target tractogram.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  --out_name OUT_NAME   Filename of the transformation matrix,
                        the registration type will be appended as a suffix,
                        [<out_name>_<affine/rigid>.txt]
  --only_rigid          Will only use a rigid transformation, uses affine by default.
  --moving_tractogram_ref MOVING_TRACTOGRAM_REF
                        Reference anatomy for moving_tractogram (if tck/vtk/fib/dpy␣
↪file
                        support (.nii or .nii.gz).
  --static_tractogram_ref STATIC_TRACTOGRAM_REF
                        Reference anatomy for static_tractogram (if tck/vtk/fib/dpy␣
↪file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

References:
[1] E. Garyfallidis, O. Ocegueda, D. Wassermann, M. Descoteaux
Robust and efficient linear registration of white-matter fascicles in the
space of streamlines, NeuroImage, Volume 117, 15 August 2015, Pages 124-140
(http://www.sciencedirect.com/science/article/pii/S1053811915003961)
```

2.124 scil_remove_invalid_streamlines.py

```
usage: __main__.py [-h] [--cut_invalid] [--remove_single_point]
                   [--remove_overlapping_points] [--threshold THRESHOLD]
                   [--no_empty] [--reference REFERENCE] [-f]
                   in_tractogram out_tractogram

Removal of streamlines that are out of the volume bounding box. In voxel space
no negative coordinate and no above volume dimension coordinate are possible.
Any streamline that do not respect these two conditions are removed.

The --cut_invalid option will cut streamlines so that their longest segment are
within the bounding box

positional arguments:
  in_tractogram         Tractogram filename. Format must be one of
                        trk, tck, vtk, fib, dpy.
  out_tractogram        Output filename. Format must be one of
                        trk, tck, vtk, fib, dpy.

optional arguments:
  -h, --help            show this help message and exit
  --cut_invalid         Cut invalid streamlines rather than removing them.
                        Keep the longest segment only.
  --remove_single_point
                        Consider single point streamlines invalid.
  --remove_overlapping_points
                        Consider streamlines with overlapping points invalid.
  --threshold THRESHOLD
                        Maximum distance between two points to be considered␣
↪overlapping [0.001 mm].
```

(continues on next page)

(continued from previous page)

```
  --no_empty            Do not save empty tractogram.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.125 scil_remove_labels.py

```
usage: __main__.py [-h] -i INDICES [INDICES ...] [--background BACKGROUND]
                   [-f]
                   in_labels out_labels

    Script to remove specific labels from an atlas volume.

    >>> scil_remove_labels.py DKT_labels.nii out_labels.nii.gz -i 5001 5002

positional arguments:
  in_labels             Input labels volume.
  out_labels            Output labels volume.

optional arguments:
  -h, --help            show this help message and exit
  -i INDICES [INDICES ...], --indices INDICES [INDICES ...]
                        List of labels indices to remove.
  --background BACKGROUND
                        Integer used for removed labels [0].
  -f                    Force overwriting of the output files.

    References:
        [1] Al-Sharif N.B., St-Onge E., Vogel J.W., Theaud G.,
            Evans A.C. and Descoteaux M. OHBM 2019.
            Surface integration for connectome analysis in age prediction.
```

2.126 scil_remove_outliers_ransac.py

```
usage: __main__.py [-h] [--min_fit MIN_FIT] [--max_iter MAX_ITER]
                   [--fit_thr FIT_THR] [-v] [-f]
                   in_image out_image

Remove outliers from image using the RANSAC algorithm.
The RANSAC algorithm parameters are sensitive to the input data.

NOTE: Current default parameters are tuned for ad/md/rd images only.

positional arguments:
  in_image              Nifti image.
  out_image             Corrected Nifti image.

optional arguments:
  -h, --help            show this help message and exit
  --min_fit MIN_FIT     The minimum number of data values required to fit the model.␣
↪[50]
```

(continues on next page)

(continued from previous page)

```
  --max_iter MAX_ITER  The maximum number of iterations allowed in the algorithm.␣
↪[1000]
  --fit_thr FIT_THR    Threshold value for determining when a data point fits a model.
↪ [0.01]
  -v                   If set, produces verbose output.
  -f                   Force overwriting of the output files.
```

2.127 scil_remove_similar_streamlines.py

2.128 scil_reorder_connectivity.py

```
usage: __main__.py [-h]
                   (--in_ordering IN_ORDERING | --optimal_leaf_ordering OUT_FILE)
                   [--out_suffix OUT_SUFFIX] [--out_dir OUT_DIR]
                   [--labels_list LABELS_LIST] [-f]
                   in_matrices [in_matrices ...]

Re-order one or many connectivity matrices using a text file format.
The first row are the (x) and the second row the (y), must be space separated.
The resulting matrix does not have to be square (support unequal number of
x and y).

The values refers to the coordinates (starting at 0) in the matrix, but if the
--labels_list parameter is used, the values will refers to the label which will
be converted to the appropriate coordinates. This file must be the same as the
one provided to the scil_decompose_connectivity.py

To subsequently use scil_visualize_connectivity.py with a lookup table, you
must use a label-based reording json and use --labels_list.

You can also use the Optimal Leaf Ordering(OLO) algorithm to transform a
sparse matrix into an ordering that reduces the matrix bandwidth. The output
file can then be re-used with --in_ordering. Only one input can be used with
this option, we recommand an average streamline count or volume matrix.

positional arguments:
  in_matrices           Connectivity matrices in .npy or .txt format.

optional arguments:
  -h, --help            show this help message and exit
  --in_ordering IN_ORDERING
                        Txt file with the first row as x and second as y.
  --optimal_leaf_ordering OUT_FILE
                        Output a text file with an ordering that alignsstructures␣
↪along the diagonal.
  --out_suffix OUT_SUFFIX
                        Suffix for the output matrix filename.
  --out_dir OUT_DIR     Output directory for the re-ordered matrices.
  --labels_list LABELS_LIST
                        List saved by the decomposition script,
                        --in_ordering must contain labels rather than coordinates (.
↪txt).
  -f                    Force overwriting of the output files.
```

(continues on next page)

(continued from previous page)

```
[1] Rubinov, Mikail, and Olaf Sporns. "Complex network measures of brain
    connectivity: uses and interpretations." Neuroimage 52.3 (2010):
    1059-1069.
```

2.129 scil_reorder_dwi_philips.py

```
usage: __main__.py [-h] [-f] [-v] in_dwi in_bvec in_bval in_table out_basename

Re-order gradient according to original table

positional arguments:
  in_dwi         Input dwi file.
  in_bvec        Input bvec FSL format.
  in_bval        Input bval FSL format.
  in_table       original table - first line is skipped.
  out_basename   Basename output file.

optional arguments:
  -h, --help     show this help message and exit
  -f             Force overwriting of the output files.
  -v             If set, produces verbose output.
```

2.130 scil_resample_bvals.py

```
usage: __main__.py [-h] [--tolerance INT] [-v] [-f]
                   in_bval bvals-to-extract [bvals-to-extract ...] out_bval

Resample b-values on specific b-value shells for sampling schemes where
b-values of a shell are not all identical. The --tolerance
argument needs to be adjusted to vary the accepted interval around the
targetted b-value.

For example, a b-value of 2000 and a tolerance of 20 will resample all
volumes with a b-values from 1980 to 2020 to the value of 2000.

scil_resample_bvals.py bvals 0 1000 2000 newbvals --tolerance 20

positional arguments:
  in_bval                The b-values in FSL format.
  bvals-to-extract       The list of b-values to extract. For example 0 1000 2000.
  out_bval               The name of the output b-values.

optional arguments:
  -h, --help             show this help message and exit
  --tolerance INT, -t INT
                         The tolerated gap between the b-values to extract
                         and the actual b-values.
  -v                     If set, produces verbose output.
  -f                     Force overwriting of the output files.
```

2.131 scil_resample_streamlines.py

```
usage: __main__.py [-h]
                   (--nb_pts_per_streamline NB_PTS_PER_STREAMLINE | --step_size STEP_
↪SIZE)
                   [--reference REFERENCE] [-f]
                   in_tractogram out_tractogram

Script to resample a set of streamlines to either a new number of points per
streamline or to a fixed step size. WARNING: data_per_point is not carried.

positional arguments:
  in_tractogram         Streamlines input file name.
  out_tractogram        Streamlines output file name.

optional arguments:
  -h, --help            show this help message and exit
  --nb_pts_per_streamline NB_PTS_PER_STREAMLINE
                        Number of points per streamline in the output.
  --step_size STEP_SIZE
                        Step size in the output (in mm).
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.132 scil_resample_tractogram.py

```
usage: __main__.py [-h] [--never_upsample]
                   [--point_wise_std POINT_WISE_STD | --streamline_wise_std␣
↪STREAMLINE_WISE_STD]
                   [--gaussian SIGMA | --spline SIGMA NB_CTRL_POINT]
                   [--keep_invalid_streamlines] [--downsample_per_cluster]
                   [--qbx_thresholds t [t ...]] [--seed SEED]
                   [--reference REFERENCE] [-f] [-v]
                   in_tractogram nb_streamlines out_tractogram

Script to resample a tractogram to a set number of streamlines.
Default behavior:
- IF number of requested streamlines is lower than streamline count: DOWNSAMPLE
- IF number of requested streamlines is higher than streamline count: UPSAMPLE
To prevent upsample if not desired use --never_upsample.

Can be useful to build training sets for machine learning algorithms, to
upsample under-represented bundles or downsample over-represented bundles.

Works by either selecting a subset of streamlines or by generating new
streamlines by adding gaussian noise to existing ones.

Upsampling:
    Includes smoothing to compensate for the noisiness of new streamlines
    generated by the process.
Downsampling:
    Includes the possibility of choosing randomly *per Quickbundle cluster* to
```

(continues on next page)

(continued from previous page)

```
    ensure that all clusters are represented in the final tractogram.

Example usage:
$ scil_resample_tractogram.py input.trk 1000 output.trk --point_wise_std 0.5 --spline␣
↪5 10 --keep_invalid_streamlines
$ scil_visualize_bundles.py output.trk --local_coloring --width=0.1

positional arguments:
  in_tractogram         Input tractography file.
  nb_streamlines        Number of streamlines to resample the tractogram to.
  out_tractogram        Output tractography file.

optional arguments:
  -h, --help            show this help message and exit
  --never_upsample      Make sure to never upsample a tractogram.
                        Useful when downsample batch of files using bash.
  --seed SEED           Use a specific random seed for the resampling.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Upsampling params:
  --point_wise_std POINT_WISE_STD
                        Noise to add to existing streamlines' points to generate new␣
↪ones.
  --streamline_wise_std STREAMLINE_WISE_STD
                        Noise to add to existing whole streamlines to generate new␣
↪ones.
  --gaussian SIGMA      Sigma for smoothing. Use the value of surronding X,Y,Z points␣
↪on
                        the streamline to blur the streamlines. A good sigma choice␣
↪would
                        be around 5.
  --spline SIGMA NB_CTRL_POINT
                        Sigma and number of points for smoothing. Models each␣
↪streamline
                        as a spline. A good sigma choice would be around 5 and control
                        points around 10.
  --keep_invalid_streamlines
                        Keep invalid newly generated streamlines that may go out of␣
↪the
                        bounding box.

Downsampling params:
  --downsample_per_cluster
                        If set, downsampling will be done per cluster (computed with
                        Quickbundles) to ensure that at least some streamlines are
                        kept per bundle. Else, random downsampling is performed␣
↪(default).
  --qbx_thresholds t [t ...]
                        If you chose option '--downsample_per_cluster', you may set
                        the QBx threshold value(s) here. Default: [40, 30, 20]
```

2.133 scil_resample_volume.py

```
usage: __main__.py [-h]
                   (--ref REF | --volume_size VOLUME_SIZE [VOLUME_SIZE ...] | --voxel_
↪size VOXEL_SIZE [VOXEL_SIZE ...] | --iso_min)
                   [--interp {nn,lin,quad,cubic}] [--enforce_dimensions] [-v]
                   [-f]
                   in_image out_image

Script to resample a dataset to match the resolution of another
reference dataset or to the resolution specified as in argument.

positional arguments:
  in_image              Path of the input volume.
  out_image             Path of the resampled volume.

optional arguments:
  -h, --help            show this help message and exit
  --ref REF             Reference volume to resample to.
  --volume_size VOLUME_SIZE [VOLUME_SIZE ...]
                        Sets the size for the volume. If the value is set to is Y, it␣
↪will resample to a shape of Y x Y x Y.
  --voxel_size VOXEL_SIZE [VOXEL_SIZE ...]
                        Sets the voxel size. If the value is set to is Y, it will set␣
↪a voxel size of Y x Y x Y.
  --iso_min             Resample the volume to R x R x R with R being the smallest␣
↪current voxel dimension.
  --interp {nn,lin,quad,cubic}
                        Interpolation mode.
                        nn: nearest neighbour
                        lin: linear
                        quad: quadratic
                        cubic: cubic
                        Defaults to linear
  --enforce_dimensions  Enforce the reference volume dimension.
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.134 scil_reshape_to_reference.py

```
usage: __main__.py [-h] [--interpolation {linear,nearest}] [--keep_dtype] [-f]
                   in_file in_ref_file out_file

Reshape / reslice / resample *.nii or *.nii.gz using a reference.
This script can be used to align freesurfer/civet output, as .mgz,
to the original input image.

>>> scil_reshape_to_reference.py wmparc.mgz t1.nii.gz wmparc_t1.nii.gz \
    --interpolation nearest

positional arguments:
  in_file               Path of the image (.nii or .mgz) to be reshaped.
  in_ref_file           Path of the reference image (.nii).
  out_file              Output filename of the reshaped image (.nii).
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  --interpolation {linear,nearest}
                        Interpolation: "linear" or "nearest". [linear]
  --keep_dtype          If True, keeps the data_type of the input image (in_file)␣
↪when saving the output image (out_file).
  -f                    Force overwriting of the output files.
```

2.135 scil_run_commit.py

```
usage: __main__.py [-h] [--b_thr B_THR] [--nbr_dir NBR_DIR]
                   [--nbr_iter NBR_ITER] [--in_peaks IN_PEAKS]
                   [--in_tracking_mask IN_TRACKING_MASK] [--commit2]
                   [--lambda_commit_2 LAMBDA_COMMIT_2] [--ball_stick]
                   [--para_diff PARA_DIFF]
                   [--perp_diff PERP_DIFF [PERP_DIFF ...]]
                   [--iso_diff ISO_DIFF [ISO_DIFF ...]]
                   [--keep_whole_tractogram]
                   [--save_kernels DIRECTORY | --load_kernels DIRECTORY]
                   [--compute_only] [--processes NBR] [-f] [-v]
                   in_tractogram in_dwi in_bval in_bvec out_dir

Convex Optimization Modeling for Microstructure Informed Tractography (COMMIT)
estimates, globally, how a given tractogram explains the DWI in terms of signal
fit, assuming a certain forward microstructure model. It assigns a weight to
each streamline, which represents how well it explains the DWI signal globally.
The default forward microstructure model is stick-zeppelin-ball, which requires
multi-shell data and a peak file (principal fiber directions in each voxel,
typically from a field of fODFs).

It is possible to use the ball-and-stick model for single-shell and multi-shell
data. In this case, the peak file is not mandatory. Multi-shell should follow a
"NODDI protocol" (low and high b-values), multiple shells with similar b-values
should not be used with COMMIT.

The output from COMMIT is:
- fit_NRMSE.nii.gz
    fiting error (Normalized Root Mean Square Error)
- fit_RMSE.nii.gz
    fiting error (Root Mean Square Error)
- results.pickle
    Dictionary containing the experiment parameters and final weights
- compartment_EC.nii.gz (est. Extra-Cellular signal fraction)
- compartment_IC.nii.gz (est. Intra-Cellular signal fraction)
- compartment_ISO.nii.gz (est. isotropic signal fraction (freewater comportment))
    Each of COMMIT compartments
- streamline_weights.txt
    Text file containing the commit weights for each streamline of the
    input tractogram.
- streamlines_length.txt
    Text file containing the length (mm) of each streamline
- tot_streamline_weights
    Text file containing the total commit weights of each streamline.
```

(continues on next page)

(continued from previous page)

```
    Equal to commit_weights * streamlines_length (W_i * L_i)
- essential.trk / non_essential.trk
    Tractograms containing the streamlines below or equal (essential) and
    above (non_essential) a threshold_weights of 0.

This script can divide the input tractogram in two using a threshold to apply
on the streamlines' weight. The threshold used is 0.0, keeping only
streamlines that have non-zero weight and that contribute to explain the DWI
signal. Streamlines with 0 weight are essentially not necessary according to
COMMIT.

COMMIT2 is available only for HDF5 data from scil_decompose_connectivity.py and
with the --ball_stick option. Use the --commit2 option to activite it, slightly
longer computation time. This wrapper offers a simplify way to call COMMIT, but
does not allow to use (or fine-tune) every parameters. If you want to use COMMIT
with full access to all parameters, visit: https://github.com/daducci/COMMIT

When tunning parameters, such as --iso_diff, --para_diff, --perp_diff or
--lambda_commit_2 you should evaluate the quality of results by:
    - Looking at the 'density' (GTM) of the connnectome (essential tractogram)
    - Confirm the quality of WM bundles reconstruction (essential tractogram)
    - Inspect the (N)RMSE map and look for peaks or anomalies
    - Compare the density map before and after (essential tractogram)

positional arguments:
  in_tractogram         Input tractogram (.trk or .tck or .h5).
  in_dwi                Diffusion-weighted images used by COMMIT (.nii.gz).
  in_bval               b-values in the FSL format (.bval).
  in_bvec               b-vectors in the FSL format (.bvec).
  out_dir               Output directory for the COMMIT maps.

optional arguments:
  -h, --help            show this help message and exit
  --b_thr B_THR         Limit value to consider that a b-value is on an existing␣
↪shell.
                        Above this limit, the b-value is placed on a new shell. This␣
↪includes b0s values.
  --nbr_dir NBR_DIR     Number of directions, on the half of the sphere,
                        representing the possible orientations of the response␣
↪functions [500].
  --nbr_iter NBR_ITER   Maximum number of iterations [1000].
  --in_peaks IN_PEAKS   Peaks file representing principal direction(s) locally,
                         typically coming from fODFs. This file is mandatory for the␣
↪default
                         stick-zeppelin-ball model.
  --in_tracking_mask IN_TRACKING_MASK
                        Binary mask where tratography was allowed.
                        If not set, uses a binary mask computed from the streamlines.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

COMMIT2 options:
  --commit2             Run commit2, requires .h5 as input and will force
                        ball&stick model.
  --lambda_commit_2 LAMBDA_COMMIT_2
```

(continues on next page)

(continued from previous page)

```
                        Specify the clustering prior strength [0.001].

Model options:
  --ball_stick          Use the ball&Stick model, disable the zeppelin compartment.
                        Only model suitable for single-shell data.
  --para_diff PARA_DIFF
                        Parallel diffusivity in mm^2/s.
                        Default for ball_stick: 1.7E-3
                        Default for stick_zeppelin_ball: 1.7E-3
  --perp_diff PERP_DIFF [PERP_DIFF ...]
                        Perpendicular diffusivity in mm^2/s.
                        Default for ball_stick: None
                        Default for stick_zeppelin_ball: [0.51E-3]
  --iso_diff ISO_DIFF [ISO_DIFF ...]
                        Istropic diffusivity in mm^2/s.
                        Default for ball_stick: [2.0E-3]
                        Default for stick_zeppelin_ball: [1.7E-3, 3.0E-3]

Tractogram options:
  --keep_whole_tractogram
                        Save a tractogram copy with streamlines weights in the data_
↪per_streamline
                        [False].
  --compute_only        Compute kernels only, --save_kernels must be used.

Kernels options:
  --save_kernels DIRECTORY
                        Output directory for the COMMIT kernels.
  --load_kernels DIRECTORY
                        Input directory where the COMMIT kernels are located.

References:
[1] Daducci, Alessandro, et al. "COMMIT: convex optimization modeling for
    microstructure informed tractography." IEEE transactions on medical
    imaging 34.1 (2014): 246-257.
[2] Schiavi, Simona, et al. "A new method for accurate in vivo mapping of
    human brain connections using microstructural and anatomical information."
    Science advances 6.31 (2020): eaba8245.
```

2.136 scil_run_nlmeans.py

```
usage: __main__.py [-h] [--mask] [--sigma float] [--log LOGFILE]
                   [--processes NBR] [-v] [-f]
                   in_image out_image number_coils

Script to denoise a dataset with the Non Local Means algorithm.

positional arguments:
  in_image          Path of the image file to denoise.
  out_image         Path to save the denoised image file.
  number_coils      Number of receiver coils of the scanner.
                    Use number_coils=1 in the case of a SENSE (GE, Philips)␣
↪reconstruction and
                    number_coils >= 1 for GRAPPA reconstruction (Siemens). number_
↪coils=4 works well for the 1.5T
```

(continues on next page)

(continued from previous page)

```
                    in Sherbrooke. Use number_coils=0 if the noise is considered␣
↪Gaussian distributed.

optional arguments:
  -h, --help            show this help message and exit
  --mask                Path to a binary mask. Only the data inside the mask will be used␣
↪for computations
  --sigma float         The standard deviation of the noise to use instead of computing ␣
↪it automatically.
  --log LOGFILE         If supplied, name of the text file to store the logs.
  --processes NBR       Number of sub-processes to start.
                        Default: [1]
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.137 scil_save_connections_from_hdf5.py

```
usage: __main__.py [-h] [--include_dps] [--edge_keys EDGE_KEYS [EDGE_KEYS ...]
                   | --node_keys NODE_KEYS [NODE_KEYS ...]] [--save_empty]
                   [--labels_list LABELS_LIST] [-f]
                   in_hdf5 out_dir

Save individual connection of an hd5f from scil_decompose_connectivity.py.
Useful for quality control and visual inspections.

It can either save all connections, individual connections specified with
edge_keys or connections from specific nodes with node_keys.

With the option save_empty, a label_lists, as a txt file, must be provided.
This option saves existing connections and empty connections.

The output is a directory containing the thousands of connections:
out_dir/
    |-- LABEL1_LABEL1.trk
    |-- LABEL1_LABEL2.trk
    |-- [...]
    |-- LABEL90_LABEL90.trk

positional arguments:
  in_hdf5               HDF5 filename (.h5) containing decomposed connections.
  out_dir               Path of the output directory.

optional arguments:
  -h, --help            show this help message and exit
  --include_dps         Include the data_per_streamline the metadata.
  --edge_keys EDGE_KEYS [EDGE_KEYS ...]
                        Keys to identify the edges of interest (LABEL1_LABEL2).
  --node_keys NODE_KEYS [NODE_KEYS ...]
                        Node keys to identify the sub-network of interest.
  --save_empty          Save empty connections.
  --labels_list LABELS_LIST
                        A txt file containing a list saved by the decomposition␣
↪script.
  -f                    Force overwriting of the output files.
```

2.138 scil_score_bundles.py

```
usage: __main__.py [-h] [--json_prefix p] [--gt_dir DIR] [--indent INDENT]
                   [--sort_keys] [-f] [--reference REFERENCE] [-v]
                   [--no_bbox_check]
                   gt_config bundles_dir

This script is similar to scil_score_tractogram, but it supposes that the
bundles are already segmented, and saved as follow:
    main_dir/
        segmented_VB/*_VS.trk.
        segmented_IB/*_*_IC.trk   (optional)
        segmented_WPC/*_wpc.trk  (optional)
        IS.trk  OR  NC.trk  (if segmented_IB is present)

Config file
-----------

The config file needs to be a json containing a dict of the ground-truth
bundles as keys. The value for each bundle is itself a dictionnary with:

    - gt_mask: expected result. OL and OR metrics will be computed from this.*

* Files must be .tck, .trk, .nii or .nii.gz. If it is a tractogram, a mask will
be created. If it is a nifti file, it will be considered to be a mask.

Exemple config file:
{
  "Ground_truth_bundle_0": {
    "gt_mask": "PATH/bundle0.nii.gz",
  }
}

Tractometry
-----------

Global connectivity metrics:
    Computed by default:
    - VS: valid streamlines, belonging to a bundle (i.e. respecting all the
        criteria for that bundle; endpoints, limit_mask, gt_mask.).
    - IS: invalid streamlines. All other streamlines. IS = IC + NC.

    Optional:
    - WPC: wrong path connections, streamlines connecting correct ROIs but not
        respecting the other criteria for that bundle. Such streamlines always
        exist but they are only saved separately if specified in the options.
        Else, they are merged back with the IS.
        ** By definition. WPC are only computed if "limits masks" are provided.
    - IC: invalid connections, streamlines joining an incorrect combination of
        ROIs. Use carefully, quality depends on the quality of your ROIs and no
        analysis is done on the shape of the streamlines.
    - NC: no connections. Invalid streamlines minus invalid connections.

Fidelity metrics:
    - OL : Overlap. Percentage of ground truth voxels containing streamline(s)
        for a given bundle.
    - OR: Overreach. Amount of voxels containing streamline(s) when they
```

(continues on next page)

(continued from previous page)

```
        shouldn't, for a given bundle. We compute two versions :
        OR_pct_vs = divided by the total number of voxel covered by the bundle.
           (percentage of the voxels touched by VS).
           Values range between 0 and 100%. Values are not defined when we
           recovered no streamline for a bundle, but we set the OR_pct_vs to 0
           in that case.
        OR_pct_gt = divided by the total size of the ground truth bundle mask.
           Values could be higher than 100%.
    - f1 score (which is the same as the Dice score).

positional arguments:
  gt_config             .json dict configured as specified above.
  bundles_dir           Directory containing all bundles.
                        (Ex: Output directory for scil_score_tractogram).
                        It is expected to contain a file IS.trk and
                        files segmented_VB/*_VS.trk, with, possibly, files
                        segmented_WPC/*_wpc.trk and segmented_IC/

optional arguments:
  -h, --help            show this help message and exit
  --json_prefix p       Prefix of the output json file. Ex: 'study_x_'.
                        Suffix will be results.json. File will be saved inside␣
↪bundles_dir.
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).

Additions to gt_config:
  --gt_dir DIR          Root path of the ground truth files listed in the gt_config.
                        If not set, filenames in the config file are considered
                        as absolute paths.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.139 scil_score_tractogram.py

```
usage: __main__.py [-h] [--json_prefix p] [--gt_dir DIR]
                   [--use_gt_masks_as_all_masks] [--dilate_endpoints NB_PASS]
                   [--remove_invalid] [--save_wpc_separately] [--compute_ic]
                   [--unique] [--remove_wpc_belonging_to_another_bundle]
                   [--no_empty] [--indent INDENT] [--sort_keys] [-f]
                   [--reference REFERENCE] [-v] [--no_bbox_check]
                   in_tractogram gt_config out_dir

Scores input tractogram overall and bundlewise.

Outputs
```

(continues on next page)

(continued from previous page)

```
-------

    - results.json: Contains a full tractometry report.
    - processing_stats.json: Contains information on the segmentation of
    bundles (ex: the number of wpc per criteria).
    - Splits the input tractogram into
        segmented_VB/*_VS.trk.
        segmented_IB/*_*_IC.trk   (if args.compute_ic)
        segmented_WPC/*_wpc.trk  (if args.save_wpc_separately)
        IS.trk     OR      NC.trk  (if args.compute_ic)

By default, if a streamline fits in many bundles, it will be included in every
one. This means a streamline may be a VS for a bundle and an IS for
(potentially many) others. If you want to assign each streamline to at most one
bundle, use the `--unique` flag.

Config file
-----------

The config file needs to be a json containing a dict of the ground-truth
bundles as keys. The value for each bundle is itself a dictionnary with:

Mandatory:
    - endpoints OR [head AND tail]: filename for the endpoints ROI.
        If 'enpoints' is used, we will automatically separate the mask into two
        ROIs, acting as head and tail. Quality check is strongly recommended.

Optional:
    Concerning metrics:
    - gt_mask: expected result. OL and OR metrics will be computed from this.*

    Concerning inclusion criteria (other streamlines will be WPC):
    - all_mask: ROI serving as "all" criteria: to be included in the bundle,
        ALL points of a streamline must be inside the mask.*
    - any_mask: ROI serving as "any" criteria: streamlines
        must touch that mask in at least one point ("any" point) to be included
        in the bundle.
    - angle: angle criteria. Streamlines containing loops and sharp turns above
        given angle will be rejected from the bundle.
    - length: maximum and minimum lengths per bundle.
    - length_x / length_x_abs: maximum and minimum total distance in the x
        direction (i.e. first coordinate).**
    - length_y / length_y_abs: maximum and minimum total distance in the y
        direction (i.e. second coordinate).**
    - length_z / length_z_abs: maximum and minimum total distance in the z
        direction (i.e. third coordinate).**

* Files must be .tck, .trk, .nii or .nii.gz. If it is a tractogram, a mask will
be created. If it is a nifti file, it will be considered to be a mask.
** With absolute values: coming back on yourself will contribute to the total
distance instead of cancelling it.

Exemple config file:
{
  "Ground_truth_bundle_0": {
    "gt_mask": "PATH/bundle0.nii.gz",
    "angle": 300,
```

(continues on next page)

(continued from previous page)

```
    "length": [140, 150],
    "endpoints": "PATH/file1.nii.gz"
  }
}

Tractometry
-----------

Global connectivity metrics:
    Computed by default:
    - VS: valid streamlines, belonging to a bundle (i.e. respecting all the
        criteria for that bundle; endpoints, limit_mask, gt_mask.).
    - IS: invalid streamlines. All other streamlines. IS = IC + NC.

    Optional:
    - WPC: wrong path connections, streamlines connecting correct ROIs but not
        respecting the other criteria for that bundle. Such streamlines always
        exist but they are only saved separately if specified in the options.
        Else, they are merged back with the IS.
        ** By definition. WPC are only computed if "limits masks" are provided.
    - IC: invalid connections, streamlines joining an incorrect combination of
        ROIs. Use carefully, quality depends on the quality of your ROIs and no
        analysis is done on the shape of the streamlines.
    - NC: no connections. Invalid streamlines minus invalid connections.

Fidelity metrics:
    - OL : Overlap. Percentage of ground truth voxels containing streamline(s)
        for a given bundle.
    - OR: Overreach. Amount of voxels containing streamline(s) when they
        shouldn't, for a given bundle. We compute two versions :
        OR_pct_vs = divided by the total number of voxel covered by the bundle.
           (percentage of the voxels touched by VS).
           Values range between 0 and 100%. Values are not defined when we
           recovered no streamline for a bundle, but we set the OR_pct_vs to 0
           in that case.
        OR_pct_gt = divided by the total size of the ground truth bundle mask.
           Values could be higher than 100%.
    - f1 score (which is the same as the Dice score).

positional arguments:
  in_tractogram         Input tractogram to score
  gt_config             .json dict configured as specified above.
  out_dir               Output directory for the resulting segmented bundles.

optional arguments:
  -h, --help            show this help message and exit
  --json_prefix p       Prefix of the two output json files. Ex: 'study_x_'.Files␣
↪will be saved inside out_dir.
                        Suffixes will be 'processing_stats.json' and 'results.json'.
  --no_empty            Do not write file if there is no streamline.
  -f                    Force overwriting of the output files.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
```

(continues on next page)

(continued from previous page)

```
                        tractograms (ignores the presence of invalid streamlines).

Additions to gt_config:
  --gt_dir DIR          Root path of the ground truth files listed in the gt_config.
                        If not set, filenames in the config file are considered
                        as absolute paths.
  --use_gt_masks_as_all_masks
                        If set, the gt_config's 'gt_mask' will also be used as
                        'all_mask' for each bundle. Note that this means the
                        OR will necessarily be 0.

Preprocessing:
  --dilate_endpoints NB_PASS
                        Dilate endpoint masks n-times. Default: 0.
  --remove_invalid      Remove invalid streamlines before scoring.

Tractometry choices:
  --save_wpc_separately
                        If set, streamlines rejected from VC based on the config
                        file criteria will be saved separately from IS (and IC)
                        in one file *_wpc.tck per bundle.
  --compute_ic          If set, IS are split into NC + IC, where IC are computed as␣
↪one bundle per
                        pair of ROI not belonging to a true connection, named
                        *_*_IC.tck.
  --unique              If set, streamlines are assigned to the first bundle they fit␣
↪in and not to all.
  --remove_wpc_belonging_to_another_bundle
                        If set, WPC actually belonging to any VB (in the
                        case of overlapping ROIs) will be removed
                        from the WPC classification.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.140 scil_screenshot_dti.py

```
usage: __main__.py [-h] [--shells SHELLS [SHELLS ...]]
                   [--out_suffix OUT_SUFFIX] [--out_dir OUT_DIR] [-f]
                   in_dwi in_bval in_bvec in_template

Register DWI to a template for screenshots.
The templates are on http://www.bic.mni.mcgill.ca/ServicesAtlases/ICBM152NLin2009

For quick quality control, the MNI template can be downsampled to 2mm iso.
Axial, coronal and sagittal slices are captured.

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bval file, in FSL format.
  in_bvec               Path of the bvec file, in FSL format.
  in_template           Path to the target MNI152 template for registration,
                        use the one provided online.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  --shells SHELLS [SHELLS ...]
                        Shells to use for DTI fit (usually below 1200), b0 must be␣
↪listed.
  --out_suffix OUT_SUFFIX
                        Add a suffix to the output, else the axis name is used.
  --out_dir OUT_DIR     Put all images in a specific directory.
  -f                    Force overwriting of the output files.
```

2.141 scil_screenshot_volume_mosaic_overlap.py

```
usage: __main__.py [-h] [--in_labelmap IN_LABELMAP]
                   [--axis_name {axial,coronal,sagittal}]
                   [--overlap_factor OVERLAP_HORIZ OVERLAP_VERT]
                   [--win_dims WIDTH HEIGHT] [--vol_cmap_name VOL_CMAP_NAME]
                   [--labelmap_cmap_name LABELMAP_CMAP_NAME] [-f]
                   in_vol in_transparency_mask out_fname slice_ids
                   [slice_ids ...] mosaic_rows_cols mosaic_rows_cols

Compose a mosaic of screenshots of the given image volume slices along the
requested axis. The provided transparency mask (e.g. a brain mask volume) is
used to set the screenshot values outside the mask non-zero values to full
transparency. Additionally, if a labelmap image is provided (e.g. a tissue
segmentation map), it is overlaid on the volume slices.

A labelmap image can be provided as the image volume, without requiring it as
the optional argument if only the former needs to be plot.

The screenshots are overlapped according to the given factors.

The mosaic supports either horizontal, vertical or matrix arrangements.

Example:
python scil_screenshot_volume_mosaic_overlap.py   t1.nii.gz   brain_mask.nii.gz  ␣
↪mosaic_overlap_t1.png   30 40 50 60 70 80 90 100   1 8

python scil_screenshot_volume_mosaic_overlap.py   t1.nii.gz   brain_mask.nii.gz  ␣
↪mosaic_overlap_t1_matrix_cmap.png   30 40 50 60 70 80 90 100   2 4   --overlap_
↪factor 0.6 0.5   --vol_cmap_name plasma

python scil_screenshot_volume_mosaic_overlap.py   tissue_map.nii.gz   brain_mask.nii.
↪gz   mosaic_overlap_tissue_map.png   30 40 50 60 70 80 90 100   2 4   --vol_cmap_
↪name plasma

python scil_screenshot_volume_mosaic_overlap.py   t1.nii.gz   brain_mask.nii.gz  ␣
↪mosaic_overlap_t1_tisue_map.png   30 40 50 60 70 80 90 100   2 4   --in_labelmap␣
↪tissue_map.nii.gz   --axis_name sagittal   --labelmap_cmap_name viridis

positional arguments:
  in_vol                Input volume image file.
  in_transparency_mask  Input mask image file.
  out_fname             Name of the output image mosaic (e.g. mosaic.jpg, mosaic.png).
```

(continues on next page)

(continued from previous page)

```
  slice_ids             Slice indices for the mosaic.
  mosaic_rows_cols      The mosaic row and column count.

optional arguments:
  -h, --help            show this help message and exit
  --in_labelmap IN_LABELMAP
                        Labelmap image.
  --axis_name {axial,coronal,sagittal}
                        Name of the axis to visualize. [axial]
  --overlap_factor OVERLAP_HORIZ OVERLAP_VERT
                        The overlap factor with respect to the dimension. [(0.6, 0.0)]
  --win_dims WIDTH HEIGHT
                        The dimensions for the vtk window. [(768, 768)]
  --vol_cmap_name VOL_CMAP_NAME
                        Colormap name for the volume image data. [None]
  --labelmap_cmap_name LABELMAP_CMAP_NAME
                        Colormap name for the labelmap image data. [viridis]
  -f                    Force overwriting of the output files.
```

2.142 scil_search_keywords.py

```
usage: __main__.py [-h] [--search_parser] [-v] keywords [keywords ...]

Search through all of SCILPY scripts and their docstrings. The output of the
search will be the intersection of all provided keywords, found either in the
script name or in its docstring.
By default, print the matching filenames and the first sentence of the
docstring. If --verbose if provided, print the full docstring.

Examples:
    scil_search_keywords.py tractogram filtering
    scil_search_keywords.py --search_parser tractogram filtering -v

positional arguments:
  keywords          Search the provided list of keywords.

optional arguments:
  -h, --help        show this help message and exit
  --search_parser   Search through and display the full script argparser instead of␣
↪looking only at the docstring. (warning: much slower).
  -v                If set, produces verbose output.
```

2.143 scil_set_response_function.py

```
usage: __main__.py [-h] [--no_factor] [-f] input tuple output

Replace the fiber response function in the FRF file.
Use this script when you want to use a fixed response function
and keep the mean b0.

The FRF file is obtained from scil_compute_ssst_frf.py
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  input         Path of the FRF file.
  tuple         Replace the response function with
                this fiber response function x 10**-4 (e.g. 15,4,4).
  output        Path of the new FRF file.

optional arguments:
  -h, --help    show this help message and exit
  --no_factor   If supplied, the fiber response function is
                evaluated without the x 10**-4 factor. [False].
  -f            Force overwriting of the output files.
```

2.144 scil_shuffle_streamlines.py

```
usage: __main__.py [-h] [--seed SEED] [--reference REFERENCE] [-f]
                   in_tractogram out_tractogram

Shuffle the ordering of streamlines.

positional arguments:
  in_tractogram         Input tractography file.
  out_tractogram        Output tractography file.

optional arguments:
  -h, --help            show this help message and exit
  --seed SEED           Random number generator seed [None].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.145 scil_smooth_streamlines.py

```
usage: __main__.py [-h] (--gaussian SIGMA | --spline SIGMA NB_CTRL_POINT)
                   [-e ERROR_RATE] [--reference REFERENCE] [-f] [-v]
                   in_tractogram out_tractogram

This script will smooth the streamlines, usually to remove the
'wiggles' in probabilistic tracking.
Two choices of methods are available:
- Gaussian will use the surrounding coordinates for smoothing.
Streamlines are resampled to 1mm step-size and the smoothing is
performed on the coordinate array. The sigma will be indicative of the
number of points surrounding the center points to be used for blurring.

- Spline will fit a spline curve to every streamline using a sigma and
the number of control points. The sigma represents the allowed distance
from the control points. The control points for the spline fit will be
the resampled streamline.
```

(continues on next page)

(continued from previous page)

```
This script enforces endpoints to remain the same.

WARNING:
- too low of a sigma (e.g: 1) with a lot of control points (e.g: 15)
will create crazy streamlines that could end up out of the bounding box.
- data_per_point will be lost.

positional arguments:
  in_tractogram         Input tractography file.
  out_tractogram        Output tractography file.

optional arguments:
  -h, --help            show this help message and exit
  --gaussian SIGMA      Sigma for smoothing. Use the value of surronding
                        X,Y,Z points on the streamline to blur the streamlines.
                        A good sigma choice would be around 5.
  --spline SIGMA NB_CTRL_POINT
                        Sigma for smoothing. Model each streamline as a spline.
                        A good sigma choice would be around 5 and control point␣
↪around 10.
  -e ERROR_RATE         Maximum compression distance in mm after smoothing. [0.1]
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.146 scil_smooth_surface.py

```
usage: __main__.py [-h] [-m VTS_MASK] [-n NB_STEPS] [-s STEP_SIZE] [-f] [-v]
                   in_surface out_surface

Script to smooth a surface with a Laplacian blur.

step_size from 0.1 to 10 is recommended
Smoothing_time = step_size * nb_steps
    [1, 10] for a small smoothing
    [10, 100] for a moderate smoothing
    [100, 1000] for a big smoothing

positional arguments:
  in_surface            Input surface (.vtk).
  out_surface           Output smoothed surface (.vtk).

optional arguments:
  -h, --help            show this help message and exit
  -m VTS_MASK, --vts_mask VTS_MASK
                        Vertices mask, where to apply the flow (.npy).
  -n NB_STEPS, --nb_steps NB_STEPS
                        Number of steps for laplacian smooth [2].
  -s STEP_SIZE, --step_size STEP_SIZE
                        Laplacian smooth step size [5.0].
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

(continues on next page)

(continued from previous page)

```
References:
[1] St-Onge, E., Daducci, A., Girard, G. and Descoteaux, M. 2018.
    Surface-enhanced tractography (SET). NeuroImage.
```

2.147 scil_snr_in_roi.py

```
usage: __main__.py [-h] [--noise_mask NOISE_MASK | --noise_map NOISE_MAP]
                   [--b0_thr B0_THR] [--out_basename OUT_BASENAME]
                   [--split_shells] [--indent INDENT] [--sort_keys] [-v]
                   in_dwi in_bval in_bvec in_mask

Script to compute signal to noise ratio (SNR) in a region of interest (ROI)
of a DWI volume.

It will compute the SNR for all DWI volumes of the input image seperately.
The output will contain the SNR which is the ratio of mean(signal) / std(noise).
The mean of the signal is computed inside the mask.
The standard deviation of the noise is estimated inside the noise_mask
or inside the same mask if a noise_map is provided.
If it's not supplied, it will be estimated using the data outside the brain,
computed with Dipy medotsu

If verbose is True, the SNR for every DWI volume will be outputed.

This works best in a well-defined ROI such as the corpus callosum.
It is heavily dependent on the ROI and its quality.

We highly recommend using a noise_map if you can acquire one.
See refs [1, 2] that describe the noise map acquisition.
[1] St-Jean, et al (2016). Non Local Spatial and Angular Matching... https://doi.org/
↪10.1016/j.media.2016.02.010
[2] Reymbaut, et al (2021). Magic DIAMOND... https://doi.org/10.1016/j.media.2021.
↪101988

positional arguments:
  in_dwi                Path of the input diffusion volume.
  in_bval               Path of the bvals file, in FSL format.
  in_bvec               Path of the bvecs file, in FSL format.
  in_mask               Binary mask of the region used to estimate SNR.

optional arguments:
  -h, --help            show this help message and exit
  --b0_thr B0_THR       All b-values with values less than or equal to b0_thr are␣
↪considered as b0s i.e. without diffusion weighting. [0.0]
  --out_basename OUT_BASENAME
                        Path and prefix for the various saved file.
  --split_shells        SNR will be split into shells.
  -v                    If set, produces verbose output.

Masks options:
  --noise_mask NOISE_MASK
                        Binary mask used to estimate the noise from the DWI.
  --noise_map NOISE_MAP
```

(continues on next page)

(continued from previous page)

```
                        Noise map.

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.148 scil_split_image.py

```
usage: __main__.py [-h] [-v] [-f]
                   in_dwi in_bval in_bvec out_basename split_indices
                   [split_indices ...]

Splits the DWI image at certain indices along the last dimension (b-values).
Many indices can be given at once by specifying multiple values. The splited
volumes are in the same order as in the original file. Also outputs the
corresponding .bval and .bvec files.

This script can be useful for splitting images at places where a b-value
extraction does not work. For instance, if one wants to split the x first
b-1500s from the rest of the b-1500s in an image, simply put x as an index.

positional arguments:
  in_dwi          The DW image file to split.
  in_bval         The b-values file in FSL format (.bval).
  in_bvec         The b-vectors file in FSL format (.bvec).
  out_basename    The basename of the output files. Indices number will be appended to␣
↪out_basename. For example, if split_indices were 3 10, the files would be saved as␣
↪out_basename_0_2, out_basename_3_10, out_basename_11_20, where the size of the last␣
↪dimension is 21 in this example.
  split_indices   The list of indices where to split the image. For example 3 10. This␣
↪would split the image in three parts, such as [:3], [3:10], [10:]. Indices must be␣
↪in increasing order.

optional arguments:
  -h, --help      show this help message and exit
  -v              If set, produces verbose output.
  -f              Force overwriting of the output files.
```

2.149 scil_split_tractogram.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR]
                   (--chunk_size CHUNK_SIZE | --nb_chunks NB_CHUNKS)
                   [--split_per_cluster | --do_not_randomize]
                   [--qbx_thresholds t [t ...]] [--seed SEED]
                   [--reference REFERENCE] [-f] [-v]
                   in_tractogram out_prefix

Split a tractogram into multiple files, 2 options available :
Split into X files, or split into files of Y streamlines.

By default, streamlines to add to each chunk will be chosen randomly.
```

(continues on next page)

(continued from previous page)

```
Optionally, you can split streamlines...
    - sequentially (the first n/nb_chunks streamlines in the first chunk and so
     on).
    - randomly, but per Quickbundles clusters.

positional arguments:
  in_tractogram         Tractogram input file name.
  out_prefix            Prefix for the output tractogram, index will be appended
                        automatically (ex, _0.trk), based on input type.

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Put all output tractogram in a specific directory.
  --chunk_size CHUNK_SIZE
                        The maximum number of streamlines per file.
  --nb_chunks NB_CHUNKS
                        Divide the file in equal parts.
  --split_per_cluster   If set, splitting will be done per cluster (computed with
                        Quickbundles) to ensure that at least some streamlines are
                        kept from each bundle in each chunk. Else, random splitting is
                        performed (default).
  --do_not_randomize    If set, splitting is done sequentially through the original
                        sft instead of using random indices.
  --qbx_thresholds t [t ...]
                        If you chose option '--split_per_cluster', you may set the
                        QBx threshold value(s) here. Default: [40, 30, 20]
  --seed SEED           Use a specific random seed for the subsampling.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.
```

2.150 scil_split_volume_by_ids.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR] [--out_prefix OUT_PREFIX]
                   [-r min max min max] [--background BACKGROUND] [-f]
                   in_labels

Split a label image into multiple images where the name of the output images
is the id of the label (ex. 35.nii.gz, 36.nii.gz, ...). If the --range option
is not provided, all labels of the image are extracted. The label 0 is
considered as the background and is ignored.

IMPORTANT: your label image must be of an integer type.

positional arguments:
  in_labels             Path of the input label file, in a format supported by␣
↪Nibabel.

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Put all ouptput images in a specific directory.
  --out_prefix OUT_PREFIX
```

(continues on next page)

(continued from previous page)

```
                        Prefix to be used for each output image.
  -r min max min max, --range min max min max
                        Specifies a subset of labels to split, formatted as min max.␣
↪Ex: -r 3 5 will give files _3, _4, _5.
  --background BACKGROUND
                        Background value. Will not be saved as a separate label.␣
↪Default: 0.
  -f                    Force overwriting of the output files.
```

2.151 scil_split_volume_by_labels.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR] [--out_prefix OUT_PREFIX]
                   (--scilpy_lut {freesurfer_subcortical,dk_aggregate_structures,
↪freesurfer_desikan_killiany} | --custom_lut CUSTOM_LUT)
                   [-f]
                   in_label

Split a label image into multiple images where the name of the output images
is taken from a lookup table (ex: left-lateral-occipital.nii.gz,
right-thalamus.nii.gz, ...). Only the labels included in the lookup table
are extracted.

IMPORTANT: your label image must be of an integer type.

positional arguments:
  in_label              Path of the input label file, in a format supported by␣
↪Nibabel.

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Put all ouptput images in a specific directory.
  --out_prefix OUT_PREFIX
                        Prefix to be used for each output image.
  --scilpy_lut {freesurfer_subcortical,dk_aggregate_structures,freesurfer_desikan_
↪killiany}
                        Lookup table, in the file scilpy/data/LUT, used to name the␣
↪output files.
  --custom_lut CUSTOM_LUT
                        Path of the lookup table file, used to name the output files.
  -f                    Force overwriting of the output files.
```

2.152 scil_streamlines_math.py

2.153 scil_swap_gradient_axis.py

```
usage: __main__.py [-h] (--fsl | --mrtrix) [-f]
                   gradient_sampling_file swapped_sampling_file dimension
                   [dimension ...]

Flip one or more axes of the gradient sampling matrix. It will be saved in
```

(continues on next page)

(continued from previous page)

```
the same format as input gradient sampling file.

positional arguments:
  gradient_sampling_file
                        Path to gradient sampling file. (.bvec or .b)
  swapped_sampling_file
                        Path to the swapped gradient sampling file.
  dimension             The axes you want to swap. eg: to swap the x and y axes use:␣
↪x y.

optional arguments:
  -h, --help            show this help message and exit
  --fsl                 Specify fsl format.
  --mrtrix              Specify mrtrix format.
  -f                    Force overwriting of the output files.
```

2.154 scil_tractogram_math.py

```
usage: __main__.py [-h] [--precision NBR_OF_DECIMALS] [--robust]
                   [--no_metadata] [--fake_metadata]
                   [--save_indices OUT_INDEX_FILE] [--indent INDENT]
                   [--sort_keys] [--reference REFERENCE] [-v] [-f]
                   [--no_bbox_check]
                   OPERATION INPUT_FILES [INPUT_FILES ...] OUTPUT_FILE

Performs an operation on a list of streamline files. The supported
operations are:

difference:  Keep the streamlines from the first file that are not in
               any of the following files.

intersection: Keep the streamlines that are present in all files.

union:        Keep all streamlines while removing duplicates.

concatenate:  Keep all streamlines with duplicates.

lazy_concatenate:  Keep all streamlines with duplicates, never load the whole
                    tractograms in memory. Only works with trk/tck file,
                    metadata will be lost and invalid streamlines are kept.

If a file 'duplicate.trk' have identical streamlines, calling the script using
the difference/intersection/union with a single input will remove these
duplicated streamlines.

To allow a soft match, use the --precision option to increase the allowed
threshold for similarity. A precision of 1 represents 10**(-1), so a
maximum distance of 0.1mm is allowed. If the streamlines are identical, the
default value of 3 (or 0.001mm distance) should work.

If there is a 0.5mm shift, use a precision of 0 (or 1mm distance) the --robust
option should make it work, but slightly slower.

The metadata (data per point, data per streamline) of the streamlines that
```

(continues on next page)

(continued from previous page)

```
are kept in the output will preserved. This requires that all input files
share the same type of metadata. If this is not the case, use the option
--no_metadata to strip the metadata from the output. Or --fake_metadata to
initialize dummy metadata in the file missing them.

positional arguments:
  OPERATION             The type of operation to be performed on the streamlines. Must
                        be one of the following: difference_robust, intersection_
↪robust, union_robust, difference, intersection, union, concatenate, lazy_
↪concatenate.
  INPUT_FILES           The list of files that contain the streamlines to operate on.
  OUTPUT_FILE           The file where the remaining streamlines are saved.

optional arguments:
  -h, --help            show this help message and exit
  --precision NBR_OF_DECIMALS, -p NBR_OF_DECIMALS
                        Precision used to compare streamlines [4].
  --robust, -r          Use version robust to small translation/rotation.
  --no_metadata, -n     Strip the streamline metadata from the output.
  --fake_metadata       Skip the metadata verification, create fake metadata if␣
↪missing, can lead to unexpected behavior.
  --save_indices OUT_INDEX_FILE, -s OUT_INDEX_FILE
                        Save the streamline indices to the supplied json file.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
  --no_bbox_check       Activate to ignore validity of the bounding box during␣
↪loading / saving of
                        tractograms (ignores the presence of invalid streamlines).

Json options:
  --indent INDENT       Indent for json pretty print.
  --sort_keys           Sort keys in output json.
```

2.155 scil_uniformize_streamlines_endpoints.py

```
usage: __main__.py [-h]
                   (--axis {x,y,z} | --auto | --centroid FILE | --target_roi TARGET_
↪ROI [TARGET_ROI ...])
                   [--swap] [--reference REFERENCE] [-v] [-f]
                   in_bundle out_bundle

Uniformize streamlines' endpoints according to a defined axis.
Useful for tractometry or models creation.

The --auto option will automatically calculate the main orientation.
If the input bundle is poorly defined, it is possible heuristic will be wrong.

The default is to flip each streamline so their first point's coordinate in the
defined axis is smaller than their last point (--swap does the opposite).

The --target option will use the barycenter of the target mask to define the
```

(continues on next page)

(continued from previous page)

```
axis. The target mask can be a binary mask or an atlas. If an atlas is
used, labels are expected in the form of --target atlas.nii.gz 2 3 5:7.

positional arguments:
  in_bundle             Input path of the tractography file.
  out_bundle            Output path of the uniformized file.

optional arguments:
  -h, --help            show this help message and exit
  --axis {x,y,z}        Match endpoints of the streamlines along this axis.
                        SUGGESTION: Commissural = x, Association = y, Projection = z
  --auto                Match endpoints of the streamlines along an automatically␣
↪determined axis.
  --centroid FILE       Match endpoints of the streamlines along an automatically␣
↪determined axis.
  --target_roi TARGET_ROI [TARGET_ROI ...]
                        Provide a target ROI and the labels to use.
                        Align heads to be closest to the mask barycenter.
                        If no labels are provided, all labels will be used.
  --swap                Swap head <-> tail convention. Can be useful when the␣
↪reference is not in RAS.
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.
```

2.156 scil_validate_and_correct_bvecs.py

```
usage: __main__.py [-h] [--mask MASK] [--peaks_vals PEAKS_VALS]
                   [--fa_th FA_TH] [--column_wise] [-v] [-f]
                   in_bvec in_peaks in_FA out_bvec

Detect sign flips and/or axes swaps in the gradients table from a fiber
coherence index [1]. The script takes as input the principal direction(s)
at each voxel, the b-vectors and the fractional anisotropy map and outputs
a corrected b-vectors file.

A typical pipeline could be:
>>> scil_compute_dti_metrics.py dwi.nii.gz bval bvec --not_all --fa fa.nii.gz
    --evecs peaks.nii.gz
>>> scil_validate_and_correct_bvecs.py bvec peaks_v1.nii.gz fa.nii.gz bvec_corr

Note that peaks_v1.nii.gz is the file containing the direction associated
to the highest eigenvalue at each voxel.

It is also possible to use a file containing multiple principal directions per
voxel, given that the amplitude of each direction is also given with the
argument --peaks_vals.

positional arguments:
  in_bvec               Path to bvec file.
  in_peaks              Path to peaks file.
  in_FA                 Path to the fractional anisotropy file.
```

(continues on next page)

(continued from previous page)

```
  out_bvec              Path to corrected bvec file (FSL format).

optional arguments:
  -h, --help            show this help message and exit
  --mask MASK           Path to an optional mask.
  --peaks_vals PEAKS_VALS
                        Path to peaks values file.
  --fa_th FA_TH         FA threshold. Only voxels with FA higher than fa_th will be␣
↪considered. [0.2]
  --column_wise         Specify input peaks are column-wise (..., 3, N) instead of␣
↪row-wise (..., N, 3).
  -v                    If set, produces verbose output.
  -f                    Force overwriting of the output files.

[1] Schilling KG, Yeh FC, Nath V, Hansen C, Williams O, Resnick S, Anderson AW,
Landman BA. A fiber coherence index for quality control of B-table orientation
in diffusion MRI scans. Magn Reson Imaging. 2019 May;58:82-89.
doi: 10.1016/j.mri.2019.01.018.
```

2.157 scil_validate_and_correct_eddy_gradients.py

```
usage: __main__.py [-h] [-f] in_bvec in_bval nb_dirs out_bvec out_bval

Validate and correct gradients from eddy outputs
With full AP-PA eddy outputs a full bvec bval (2x nb of dirs and bval)
that doesnt fit with the output dwi (1x nb of dir)

positional arguments:
  in_bvec     In bvec file.
  in_bval     In bval file.
  nb_dirs     Number of directions per DWI.
  out_bvec    Out bvec file.
  out_bval    Out bval file.

optional arguments:
  -h, --help  show this help message and exit
  -f          Force overwriting of the output files.
```

2.158 scil_verify_space_attributes_compatibility.py

```
usage: __main__.py [-h] in_files [in_files ...]

Will compare all input files against the first one for the compatibility
of their spatial attributes.

Spatial attributes are: affine, dimensions, voxel sizes and voxel order.

positional arguments:
  in_files    List of file to compare (trk and nii).
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help  show this help message and exit
```

2.159 scil_visualize_bingham_fit.py

```
usage: __main__.py [-h] [--slice_index SLICE_INDEX] [--win_dims WIDTH HEIGHT]
                   [--interactor {trackball,image}]
                   [--axis_name {sagittal,axial,coronal}] [--silent]
                   [--output OUTPUT] [-f]
                   [--sphere {repulsion100,repulsion200,repulsion724,symmetric362,
↪symmetric642,symmetric724}]
                   [--color_per_lobe]
                   in_bingham

Visualize 2-dimensional Bingham volume slice loaded from disk. The volume is
assumed to be saved from scil_fit_bingham_to_fodf.py.

Given an image of Bingham coefficients, this script displays a slice in a
given orientation.

positional arguments:
  in_bingham            Input SH image file.

optional arguments:
  -h, --help            show this help message and exit
  --slice_index SLICE_INDEX
                        Index of the slice to visualize along a given axis. Defaults␣
↪to middle of volume.
  --win_dims WIDTH HEIGHT
                        The dimensions for the vtk window. [(768, 768)]
  --interactor {trackball,image}
                        Specify interactor mode for vtk window. [trackball]
  --axis_name {sagittal,axial,coronal}
                        Name of the axis to visualize. [axial]
  --silent              Disable interactive visualization.
  --output OUTPUT       Path to output file.
  -f                    Force overwriting of the output files.
  --sphere {repulsion100,repulsion200,repulsion724,symmetric362,symmetric642,
↪symmetric724}
                        Name of the sphere used to reconstruct SF. [symmetric362]
  --color_per_lobe      Color each bingham distribution with a different color.␣
↪[False]
```

2.160 scil_visualize_bundles.py

```
usage: __main__.py [-h]
                   [--random_coloring SEED | --uniform_coloring R G B | --local_
↪coloring | --color_dict JSON | --color_from_streamlines KEY | --color_from_points␣
↪KEY]
                   [--shape {line,tube}] [--width WIDTH]
                   [--subsample SUBSAMPLE] [--downsample DOWNSAMPLE]
```

(continues on next page)

(continued from previous page)

```
                    [--background R G B]
                    in_bundles [in_bundles ...]

Visualize bundles.

Example usages:

# Visualize streamlines as tubes, each bundle with a different color
$ scil_visualize_bundles.py path_to_bundles/ --shape tube     --random_coloring 1337

# Visualize a tractogram with each streamlines drawn as lines, colored with
# their local orientation, but only load 1 in 10 streamlines
$ scil_visualize_bundles.py tractogram.trk --shape line --subsample 10

# Visualize CSTs as large tubes and color them from a list of colors in a file
$ scil_visualize_bundles.py path_to_bundles/CST_* --width 0.5     --color_dict colors.
↪json

positional arguments:
  in_bundles            List of tractography files supported by nibabel.

optional arguments:
  -h, --help            show this help message and exit
  --shape {line,tube}   Display streamlines either as lines or tubes.
                        [Default: tube]
  --width WIDTH         Width of tubes or lines representing streamlines
                        [Default: 0.25]
  --subsample SUBSAMPLE
                        Only load 1 in N streamlines.
                        [Default: 1]
  --downsample DOWNSAMPLE
                        Downsample streamlines to N points.
                        [Default: None]
  --background R G B    RBG values [0, 255] of the color of the background.
                        [Default: [0, 0, 0]]

Colouring options:
  --random_coloring SEED
                        Assign a random color to bundles.
  --uniform_coloring R G B
                        Assign a uniform color to streamlines.
  --local_coloring      Assign coloring to streamlines depending on their local␣
↪orientations.
  --color_dict JSON     JSON file containing colors for each bundle.
                        Bundle filenames are indicated as  keys and colors as values.
                        A 'default'  key and value can be included.
  --color_from_streamlines KEY
                        Extract a color per streamline from the data_per_streamline␣
↪property of the tractogram at the specified key.
  --color_from_points KEY
                        Extract a color per point from the data_per_point property of␣
↪the tractogram at the specified key.
```

2.161 scil_visualize_bundles_mosaic.py

```
usage: __main__.py [-h] [--uniform_coloring R G B] [--random_coloring SEED]
                   [--zoom ZOOM] [--ttf TTF] [--ttf_size TTF_SIZE]
                   [--opacity_background OPACITY_BACKGROUND]
                   [--resolution_of_thumbnails RESOLUTION_OF_THUMBNAILS]
                   [--light_screenshot] [--no_information] [--no_bundle_name]
                   [--no_streamline_number] [--reference REFERENCE] [-f]
                   in_volume in_bundles [in_bundles ...] out_image

Visualize bundles from a list. The script will output a mosaic (image) with
screenshots, 6 views per bundle in the list.

positional arguments:
  in_volume             Volume used as background (e.g. T1, FA, b0).
  in_bundles            List of tractography files supported by nibabel or binary␣
↪mask files.
  out_image             Name of the output image mosaic (e.g. mosaic.jpg, mosaic.png).

optional arguments:
  -h, --help            show this help message and exit
  --uniform_coloring R G B
                        Assign an uniform color to streamlines (or ROIs).
  --random_coloring SEED
                        Assign a random color to streamlines (or ROIs).
  --zoom ZOOM           Rendering zoom. A value greater than 1 is a zoom-in,
                        a value less than 1 is a zoom-out [1.0].
  --ttf TTF             Path of the true type font to use for legends.
  --ttf_size TTF_SIZE   Font size (int) to use for the legends [35].
  --opacity_background OPACITY_BACKGROUND
                        Opacity of background image, between 0 and 1.0 [0.4].
  --resolution_of_thumbnails RESOLUTION_OF_THUMBNAILS
                        Resolution of thumbnails used in mosaic [300].
  --light_screenshot    Keep only 3 views instead of 6 [False].
  --no_information      Don't display axis and bundle information [False].
  --no_bundle_name      Don't display bundle name [False].
  --no_streamline_number
                        Don't display bundle streamlines number [False].
  --reference REFERENCE
                        Reference anatomy for tck/vtk/fib/dpy file
                        support (.nii or .nii.gz).
  -f                    Force overwriting of the output files.
```

2.162 scil_visualize_connectivity.py

```
usage: __main__.py [-h] [--labels_list LABELS_LIST]
                   [--reorder_txt REORDER_TXT] [--lookup_table LOOKUP_TABLE]
                   [--name_axis] [--axis_text_size X_SIZE Y_SIZE]
                   [--axis_text_angle X_ANGLE Y_ANGLE] [--colormap COLORMAP]
                   [--display_legend] [--legend_min_max MIN MAX]
                   [--write_values FONT_SIZE DECIMAL] [--histogram FILENAME]
                   [--nb_bins NB_BINS] [--exclude_zeros]
                   [--chord_chart FILENAME]
                   [--percentile_threshold PERCENTILE_THRESHOLD]
```

(continues on next page)

(continued from previous page)

```
                    [--angle_threshold ANGLE_THRESHOLD] [--alpha ALPHA]
                    [--text_size TEXT_SIZE] [--text_distance TEXT_DISTANCE]
                    [--log] [--show_only] [-f]
                    in_matrix out_png

Script to display a connectivity matrix and adjust the desired visualization.
Made to work with scil_decompose_connectivity.py and
scil_reorder_connectivity.py.

This script can either display the axis labels as:
- Coordinates (0..N)
- Labels (using --labels_list)
- Names (using --labels_list and --lookup_table)
Examples of labels_list.txt and lookup_table.json can be found in the
freesurfer_flow output (https://github.com/scilus/freesurfer_flow)

If the matrix was made from a bigger matrix using scil_reorder_connectivity.py,
provide the text file(s), using --labels_list and/or --reorder_txt.

The chord chart is always displaying parting in the order they are defined
(clockwise), the color is attributed in that order following a colormap. The
thickness of the line represent the 'size/intensity', the greater the value is
the thicker the line will be. In order to hide the low values, two options are
available:
- Angle threshold + alpha, any connections with a small angle on the chord chart
    will be slightly transparent to increase the focus on bigger connections.
- Percentile, hide any connections with a value below that percentile

positional arguments:
  in_matrix             Connectivity matrix in numpy (.npy) format.
  out_png               Output filename for the connectivity matrix figure.

optional arguments:
  -h, --help            show this help message and exit
  --log                 Apply a base 10 logarithm to the matrix.
  --show_only           Do not save the figure, simply display it.
  -f                    Force overwriting of the output files.

Naming options:
  --labels_list LABELS_LIST
                        List saved by the decomposition script,
                        must contain labels rather than coordinates (.txt).
  --reorder_txt REORDER_TXT
                        File with two rows (x/y) listing the ordering (.txt).
  --lookup_table LOOKUP_TABLE
                        Lookup table with the label number as keys and the name as␣
↪values (.json).

Matplotlib options:
  --name_axis           Use the provided info/files to name axis.
  --axis_text_size X_SIZE Y_SIZE
                        Font size of the X and Y axis labels. [(10, 10)]
  --axis_text_angle X_ANGLE Y_ANGLE
                        Text angle of the X and Y axis labels. [(90, 0)]
  --colormap COLORMAP   Colormap to use for the matrix. [viridis]
  --display_legend      Display the colorbar next to the matrix.
  --legend_min_max MIN MAX
```

(continues on next page)

(continued from previous page)

```
                        Manually define the min/max of the legend.
  --write_values FONT_SIZE DECIMAL
                        Write the values at the center of each node.
                        The font size and the rouding parameters can be adjusted.

Histogram options:
  --histogram FILENAME  Compute and display/save an histogram of weights.
  --nb_bins NB_BINS     Number of bins to use for the histogram.
  --exclude_zeros       Exclude the zeros from the histogram.

Chord chart options:
  --chord_chart FILENAME
                        Compute and display/save a chord chart of weigth.
  --percentile_threshold PERCENTILE_THRESHOLD
                        Discard connections below that percentile.[0]
  --angle_threshold ANGLE_THRESHOLD
                        Angle below that theshold will be transparent.
                        Use --alpha to set opacity. Value typicallybetween 0.1 and 5␣
↪degrees. [1]
  --alpha ALPHA         Opacity for the smaller angle on the chord (0-1). [0.9]
  --text_size TEXT_SIZE
                        Size of the font for the parcels name/number [10].
  --text_distance TEXT_DISTANCE
                        Distance from the center so the parcels name/number do not␣
↪overlap
                        with the diagram [1.1].
```

2.163 scil_visualize_fodf.py

```
usage: __main__.py [-h] [--slice_index SLICE_INDEX] [--win_dims WIDTH HEIGHT]
                   [--interactor {trackball,image}]
                   [--axis_name {sagittal,axial,coronal}] [--silent]
                   [--in_transparency_mask IN_TRANSPARENCY_MASK]
                   [--output OUTPUT] [-f]
                   [--sh_basis {descoteaux07,tournier07}]
                   [--sphere {repulsion100,symmetric642,symmetric724,repulsion724,
↪repulsion200,symmetric362}]
                   [--sph_subdivide SPH_SUBDIVIDE] [--mask MASK]
                   [--colormap COLORMAP | --color_rgb COLOR_RGB COLOR_RGB COLOR_RGB]
                   [--scale SCALE] [--radial_scale_off] [--norm_off]
                   [--background BACKGROUND] [--bg_range MIN MAX]
                   [--bg_opacity BG_OPACITY] [--bg_offset BG_OFFSET]
                   [--bg_interpolation {linear,nearest}]
                   [--bg_color BG_COLOR BG_COLOR BG_COLOR] [--peaks PEAKS]
                   [--peaks_color PEAKS_COLOR PEAKS_COLOR PEAKS_COLOR]
                   [--peaks_width PEAKS_WIDTH]
                   [--peaks_values PEAKS_VALUES | --peaks_length PEAKS_LENGTH]
                   [--variance VARIANCE] [--variance_k VARIANCE_K]
                   [--var_color VAR_COLOR VAR_COLOR VAR_COLOR]
                   in_fodf

Visualize 2-dimensional fODF slice loaded from disk.

Given an image of SH coefficients, this script displays a slice in a
```

(continues on next page)

(continued from previous page)

```
given orientation. The user can also add a background on top of which the
fODF are to be displayed. Using a full SH basis, the script can be used to
visualize asymmetric fODF. The user can supply a peaks image to visualize
peaks on top of fODF.

If a transparency_mask is given (e.g. a brain mask), all values outside the
mask non-zero values are set to full transparency in the saved scene.

positional arguments:
  in_fodf               Input SH image file.

optional arguments:
  -h, --help            show this help message and exit
  --slice_index SLICE_INDEX
                        Index of the slice to visualize along a given axis. Defaults␣
↪to middle of volume.
  --win_dims WIDTH HEIGHT
                        The dimensions for the vtk window. [(768, 768)]
  --interactor {trackball,image}
                        Specify interactor mode for vtk window. [trackball]
  --axis_name {sagittal,axial,coronal}
                        Name of the axis to visualize. [axial]
  --silent              Disable interactive visualization.
  --in_transparency_mask IN_TRANSPARENCY_MASK
                        Input mask image file.
  --output OUTPUT       Path to output file.
  -f                    Force overwriting of the output files.
  --sh_basis {descoteaux07,tournier07}
                        Spherical harmonics basis used for the SH coefficients.
                        Must be either 'descoteaux07' or 'tournier07' [descoteaux07]:
                            'descoteaux07': SH basis from the Descoteaux et al.
                                            MRM 2007 paper
                            'tournier07'  : SH basis from the Tournier et al.
                                            NeuroImage 2007 paper.
  --sphere {repulsion100,symmetric642,symmetric724,repulsion724,repulsion200,
↪symmetric362}
                        Name of the sphere used to reconstruct SF. [symmetric724]
  --sph_subdivide SPH_SUBDIVIDE
                        Number of subdivisions for given sphere. If not supplied, use␣
↪the given sphere as is.
  --mask MASK           Optional mask file. Only fODF inside the mask are displayed.
  --colormap COLORMAP   Colormap for the ODF slicer. If None, then a RGB colormap␣
↪will be used. [None]
  --color_rgb COLOR_RGB COLOR_RGB COLOR_RGB
                        Uniform color for the ODF slicer given as RGB, scaled between␣
↪0 and 1. [None]
  --scale SCALE         Scaling factor for FODF. [0.5]
  --radial_scale_off    Disable radial scale for ODF slicer.
  --norm_off            Disable normalization of ODF slicer.

Background arguments:
  --background BACKGROUND
                        Background image file. If RGB, values must be between 0 and␣
↪255.
  --bg_range MIN MAX    The range of values mapped to range [0, 1] for background␣
↪image. [(bg.min(), bg.max())]
  --bg_opacity BG_OPACITY
```

(continues on next page)

(continued from previous page)

```
                        The opacity of the background image. Opacity of 0.0 means␣
↪transparent and 1.0 is completely visible. [1.0]
  --bg_offset BG_OFFSET
                        The offset of the background image. [0.5]
  --bg_interpolation {linear,nearest}
                        Interpolation mode for the background image. [nearest]
  --bg_color BG_COLOR BG_COLOR BG_COLOR
                        The color of the overall background, behind everything. Must␣
↪be RGB values scaled between 0 and 1. [(0, 0, 0)]

Peaks arguments:
  --peaks PEAKS         Peaks image file.
  --peaks_color PEAKS_COLOR PEAKS_COLOR PEAKS_COLOR
                        Color used for peaks, as RGB values scaled between 0 and 1.␣
↪If None, then a RGB colormap is used. [None]
  --peaks_width PEAKS_WIDTH
                        Width of peaks segments. [1.0]

Peaks scaling arguments:
  Choose between peaks values and arbitrary length.

  --peaks_values PEAKS_VALUES
                        Peaks values file.
  --peaks_length PEAKS_LENGTH
                        Length of the peaks segments. [0.65]

Variance arguments:
  For the visualization of fodf uncertainty, the variance is used as follow: mean + k␣
↪* sqrt(variance), where mean is the input fodf (in_fodf) and k is the scaling␣
↪factor (variance_k).

  --variance VARIANCE   FODF variance file.
  --variance_k VARIANCE_K
                        Scaling factor (k) for the computation of the fodf␣
↪uncertainty. [1]
  --var_color VAR_COLOR VAR_COLOR VAR_COLOR
                        Color of variance outline. Must be RGB values scaled between␣
↪0 and 1. [(1, 1, 1)]
```

2.164 scil_visualize_gradients.py

```
usage: __main__.py [-h]
                   (--in_gradient_scheme IN_GRADIENT_SCHEME [IN_GRADIENT_SCHEME ...]␣
↪| --dipy_sphere {symmetric362,symmetric642,symmetric724,repulsion724,repulsion100,
↪repulsion200})
                   [--dis-sym] [--out_basename OUT_BASENAME] [--res RES]
                   [--dis-sphere] [--dis-proj] [--plot_shells] [--same-color]
                   [--opacity OPACITY] [-f] [-v]

Vizualisation for directions on a sphere, either from a gradient sampling (i.e.
a list of b-vectors) or from a Dipy sphere.

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --in_gradient_scheme IN_GRADIENT_SCHEME [IN_GRADIENT_SCHEME ...]
                        Gradient sampling filename. (only accepts .bvec and
                        .bval together or only .b).
  --dipy_sphere {symmetric362,symmetric642,symmetric724,repulsion724,repulsion100,
↪repulsion200}
                        Dipy sphere choice.
  --dis-sym             Disable antipodal symmetry.
  --out_basename OUT_BASENAME
                        Output file name picture without extension (will be
                        png file(s)).
  --res RES             Resolution of the output picture(s).
  -f                    Force overwriting of the output files.
  -v                    If set, produces verbose output.

Enable/Disable renderings.:
  --dis-sphere          Disable the rendering of the sphere.
  --dis-proj            Disable rendering of the projection supershell.
  --plot_shells         Enable rendering each shell individually.

Rendering options.:
  --same-color          Use same color for all shell.
  --opacity OPACITY     Opacity for the shells.
```

2.165 scil_visualize_histogram.py

```
usage: __main__.py [-h] [--title TITLE] [--x_label X_LABEL] [--colors COLORS]
                   [--show_only] [-f]
                   in_metric in_mask n_bins out_png

Script to display a histogram of a metric (FA, MD, etc.) from a binary mask
(wm mask, vascular mask, ect.).
These two images must be coregister with each other.

>>> scil_visualize_histogram.py metric.nii.gz mask_bin.nii.gz 8
    out_filename_image.png

positional arguments:
  in_metric          Metric map ex : FA, MD,... .
  in_mask            Binary mask data to extract value.
  n_bins             Number of bins to use for the histogram.
  out_png            Output filename for the figure.

optional arguments:
  -h, --help         show this help message and exit
  --show_only        Do not save the figure, only display.
  -f                 Force overwriting of the output files.

Histogram options:
  --title TITLE      Use the provided info for the histogram title. [Histogram]
  --x_label X_LABEL  Use the provided info for the x axis name.
  --colors COLORS    Use the provided info for the bars color. [#0504aa]
```

2.166 scil_visualize_scatterplot.py

```
usage: __main__.py [-h] [--out_dir OUT_DIR] [--thr THR] [--not_exclude_zero]
                   [--in_bin_mask IN_BIN_MASK | --in_prob_maps IN_PROB_MAPS IN_PROB_
↪MAPS | --in_atlas IN_ATLAS]
                   [--atlas_lut ATLAS_LUT]
                   [--specific_label SPECIFIC_LABEL [SPECIFIC_LABEL ...]]
                   [--in_folder] [--title TITLE] [--x_label X_LABEL]
                   [--y_label Y_LABEL] [--label LABEL]
                   [--label_prob LABEL_PROB] [--marker MARKER]
                   [--marker_size MARKER_SIZE] [--transparency TRANSPARENCY]
                   [--dpi DPI] [--colors color1 color2] [--show_only] [-f]
                   in_x_map in_y_map out_name

Script to display scatter plot between two maps (ex. FA and MD, ihMT and MT).
By default, no mask is applied to the data.
Different options are available to mask or threshold data:
  - a binary mask
  - two probability maps, which can be used to threshold maps with
    --in_prob_maps. A same threshold is applied on these two maps (--thr).
  - parcellation, which can be used to plot values for each region of
    an atlas (--in_atlas) or a subset of regions (--specific_label).
    Atlas option required a json file (--atlas_lut) with indices and
    names of each label corresponding to the atlas as following:
    "1": "lh_A8m",
    "2": "rh_A8m",
    The numbers must be corresponding to the label indices in the json file.

Be careful, you can not use all of them at the same time.

For general scatter plot without mask:
>>> scil_visualize_scatterplot.py FA.nii.gz MD.nii.gz out_filename_image.png

For scatter plot with mask:
>>> scil_visualize_scatterplot.py FA.nii.gz MD.nii.gz out_filename_image.png
    --in_bin_mask mask_wm.nii.gz

For tissue probability scatter plot:
>>> scil_visualize_scatterplot.py FA.nii.gz MD.nii.gz out_filename_image.png
    --prob_maps wm_map.nii.gz gm_map.nii.gz

For scatter plot using atlas:
>>> scil_visualize_scatterplot.py FA.nii.gz MD.nii.gz out_filename_image.png
    --in_atlas atlas.nii.gz --atlas_lut atlas.json

>>> scil_visualize_scatterplot.py FA.nii.gz MD.nii.gz out_filename_image.png
    --in_atlas atlas.nii.gz --atlas_lut atlas.json
    --specific_label 34 67 87

positional arguments:
  in_x_map              Map in x axis, FA for example.
  in_y_map              Map in y axis, MD for example.
  out_name              Output filename for the figure without extension.

optional arguments:
  -h, --help            show this help message and exit
  --out_dir OUT_DIR     Output directory to save scatter plot.
```

(continues on next page)

(continued from previous page)

```
  --thr THR             Use to apply threshold only on probability maps (same for␣
↪both map) with --in_prob_maps option. [0.9]
  --not_exclude_zero    Keep zero value in data.
  --in_bin_mask IN_BIN_MASK
                        Binary mask. Use this option to extract x and y maps value␣
↪from specific mask or region: wm_mask or roi_mask for example.
  --in_prob_maps IN_PROB_MAPS IN_PROB_MAPS
                        Probability maps, WM and GW for example.
  --in_atlas IN_ATLAS   Path to the input atlas image.
  --show_only           Do not save the figure, only display.  Not avalaible with --
↪in_atlas option.
  -f                    Force overwriting of the output files.

Atlas options:
  --atlas_lut ATLAS_LUT
                        Path of the LUT file corresponding to atlas used to name the␣
↪regions of interest.
  --specific_label SPECIFIC_LABEL [SPECIFIC_LABEL ...]
                        Label list to use to do scatter plot. Label must␣
↪corresponding to atlas LUT file. [None]
  --in_folder           Save label plots in subfolder "Label_plots".

Scatter plot options:
  --title TITLE         Use the provided info for the title name.  [Scatter Plot]
  --x_label X_LABEL     Use the provided info for the x axis name.  [x]
  --y_label Y_LABEL     Use the provided info for the y axis name.  [y]
  --label LABEL         Use the provided info for the legend box corresponding to␣
↪mask or first probability map.  [None]
  --label_prob LABEL_PROB
                        Use the provided info for the legend box corresponding to the␣
↪second probability map.  [Threshold prob_map 2]
  --marker MARKER       Use the provided info for the marker shape. [.]
  --marker_size MARKER_SIZE
                        Use the provided info for the marker size. [15]
  --transparency TRANSPARENCY
                        Use the provided info for the point transparency. [0.4]
  --dpi DPI             Use the provided info for the dpi resolution. [300]
  --colors color1 color2
```

2.167 scil_visualize_seeds.py

```
usage: __main__.py [-h] [--save SAVE] [-f] tractogram

Visualize seeds used to generate the tractogram or bundle.
When tractography was run, each streamline produced by the tracking algorithm
saved its seeding point (its origin).

The tractogram must have been generated from scil_compute_local/pft_tracking.py
with the --save_seeds option.

positional arguments:
  tractogram   Tractogram file (must be trk)

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help   show this help message and exit
  --save SAVE  If set, save a screenshot of the result in the specified filename
  -f           Force overwriting of the output files.
```

2.168 scil_visualize_seeds_3d.py

```
usage: __main__.py [-h] [--tractogram TRACTOGRAM] [--colormap COLORMAP]
                   [--seed_opacity SEED_OPACITY]
                   [--tractogram_shape {line,tube}]
                   [--tractogram_opacity TRACTOGRAM_OPACITY]
                   [--tractogram_width TRACTOGRAM_WIDTH]
                   [--tractogram_color R G B [R G B ...]]
                   [--background R G B [R G B ...]]
                   in_seed_map

Visualize seeds as 3D points, with heatmaps corresponding to seed density

Example usages:

scil_visualize_seeds_3d.py seeds.nii.gz --tractogram tractogram.trk

positional arguments:
  in_seed_map           Seed density map.

optional arguments:
  -h, --help            show this help message and exit
  --tractogram TRACTOGRAM
                        Tractogram coresponding to the seeds.
  --colormap COLORMAP   Name of the map for the density coloring. Can be any colormap␣
↪that matplotlib offers.
                        [Default: bone]
  --seed_opacity SEED_OPACITY
                        Opacity of the contour generated.
                        [Default: 0.5]
  --tractogram_shape {line,tube}
                        Display streamlines either as lines or tubes.
                        [Default: tube]
  --tractogram_opacity TRACTOGRAM_OPACITY
                        Opacity of the streamlines.
                        [Default: 0.5]
  --tractogram_width TRACTOGRAM_WIDTH
                        Width of tubes or lines representing streamlines.
                        [Default: 0.05]
  --tractogram_color R G B [R G B ...]
                        Color for the tractogram.
  --background R G B [R G B ...]
                        RBG values [0, 255] of the color of the background.
                        [Default: [0, 0, 0]]
```

CHAPTER 3

Instructions for streamlines registration/transformation

```
scil_register_tractogram.py MOVING_FILE STATIC_FILE
```

The file outputted by this script is a 4x4 matrix (see the help for the option)

3.1 Linear transformation

If you want to apply a transformation coming from the previous script

```
scil_apply_transform_to_tractogram.py MOVING_FILE REFERENCE_FILE TRANSFORMATION␣
↪OUTPUT_NAME
```

Due to a difference in convention between image and tractogram the following script must be called using the –inverse flag if the transformation was obtained using AntsRegistration

```
scil_apply_transform_to_tractogram.py MOVING_FILE REFERENCE_FILE  0GenericAffine.mat␣
↪OUTPUT_NAME --inverse
```

3.2 Non-linear deformation

To apply a non-linear transformation from ANTS

```
scil_apply_transform_to_tractogram.py MOVING_FILE REFERENCE_FILE  0GenericAffine.mat␣
↪OUTPUT_NAME --inverse --in_deformation DEFORMATION_FILE
```

- The DEFORMATION_FILE needs to be the InverseWarp.nii.gz (very important)
- The OUTPUT_NAME is the output tractogram

3.3 Complete example

```
antsRegistrationSyNQuick.sh -d 3 -f mni_masked.nii.gz -m 100307__fa.nii.gz -t s -o to_
↪mni
scil_apply_transform_to_tractogram.py 100307__tracking.trk mni_masked.nii.gz to_
↪mni0GenericAffine.mat 100307__tracking_linear.trk --inverse
scil_apply_transform_to_tractogram.py 100307__tracking.trk mni_masked.nii.gz to_
↪mni0GenericAffine.mat 100307__tracking_nonlinear.trk --inverse --in_deformation to_
↪mni1InverseWarp.nii.gz
```

3.4 Apply back and forth tractogram transformation with the ANTS transformation

```
# The ANTS commands is MOVING->REFERENCE
antsRegistrationSyNQuick.sh -d 3 -f ${REFERENCE_NII.GZ_REF-SPACE} -m ${MOVING_NII.GZ_
↪MOV-SPACE} -t s -o to_reference_

# This will bring a tractogram from MOVING->REFERENCE
scil_apply_transform_to_tractogram.py ${MOVING_FILE_MOV-SPACE} ${REFERENCE_FILE_REF-
↪SPACE}
                                      to_reference_0GenericAffine.mat ${OUTPUT_NAME}
                                      --inverse
                                      --in_deformation to_reference_1InverseWarp.nii.
↪gz

# This will bring a tractogram from REFERENCE->MOVING
scil_apply_transform_to_tractogram.py ${MOVING_FILE_REF-SPACE} ${REFERENCE_FILE_MOV-
↪SPACE}
                                      to_reference_0GenericAffine.mat ${OUTPUT_NAME}
                                      --in_deformation to_reference_1Warp.nii.gz
                                      --reverse_operation
```

Bibliography

[Garyfallidis17] Garyfallidis et al. Recognition of white matter bundles using local and global streamline-based registration and clustering, Neuroimage, 2017.

[Garyfallidis15] Garyfallidis et al. Robust and efficient linear registration of white-matter fascicles in the space of streamlines, Neuroimage, 2015.

Python Module Index

S

scilpy.gradientsampling.gen_gradient_sampling, 1
scilpy.gradientsampling.multiple_shell_energy, 1
scilpy.gradientsampling.optimize_gradient_sampling, 4
scilpy.gradientsampling.save_gradient_sampling, 7
scilpy.image.operations, 8
scilpy.image.resample_volume, 10
scilpy.image.reslice, 11
scilpy.image.utils, 12
scilpy.io.image, 13
scilpy.io.streamlines, 13
scilpy.io.utils, 14
scilpy.preprocessing.distortion_correction, 17
scilpy.reconst.afd_along_streamlines, 19
scilpy.reconst.fodf, 19
scilpy.reconst.frf, 20
scilpy.reconst.raw_signal, 22
scilpy.reconst.utils, 23
scilpy.segment.models, 24
scilpy.segment.recobundlesx, 24
scilpy.segment.streamlines, 25
scilpy.segment.voting_scheme, 26
scilpy.tracking.tools, 27
scilpy.tractanalysis.distance_to_centroid, 29
scilpy.tractanalysis.features, 29
scilpy.tractanalysis.grid_intersections, 31
scilpy.tractanalysis.quick_tools, 31
scilpy.tractanalysis.reproducibility_measures, 31
scilpy.tractanalysis.streamlines_metrics, 34
scilpy.tractanalysis.todi, 34
scilpy.tractanalysis.todi_util, 37
scilpy.tractanalysis.tools, 38
scilpy.tractanalysis.uncompress, 39
scilpy.utils.bvec_bval_tools, 39
scilpy.utils.filenames, 43
scilpy.utils.image, 43
scilpy.utils.metrics_tools, 44
scilpy.utils.streamlines, 47
scilpy.utils.util, 48
scilpy.viz.gradient_sampling, 49
scilpy.viz.screenshot, 50

Index

A

absolute_value() (*in module scilpy.image.operations*), 8
add_b0s() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 4
add_bbox_arg() (*in module scilpy.io.utils*), 14
add_bvalue_b0() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 4
add_filename_suffix() (*in module scilpy.utils.filenames*), 43
add_force_b0_arg() (*in module scilpy.io.utils*), 14
add_json_args() (*in module scilpy.io.utils*), 14
add_overwrite_arg() (*in module scilpy.io.utils*), 14
add_processes_arg() (*in module scilpy.io.utils*), 14
add_reference_arg() (*in module scilpy.io.utils*), 15
add_sh_basis_args() (*in module scilpy.io.utils*), 15
add_sphere_arg() (*in module scilpy.io.utils*), 15
add_verbose_arg() (*in module scilpy.io.utils*), 15
addition() (*in module scilpy.image.operations*), 8
afd_and_rd_sums_along_streamlines() (*in module scilpy.reconst.afd_along_streamlines*), 19
afd_map_along_streamlines() (*in module scilpy.reconst.afd_along_streamlines*), 19
ALL (*scilpy.utils.bvec_bval_tools.B0ExtractionStrategy attribute*), 39
approximate_surface_node() (*in module scilpy.tractanalysis.reproducibility_measures*), 31
around() (*in module scilpy.image.operations*), 8
assert_fsl_options_exist() (*in module scilpy.io.utils*), 15
assert_gradients_filenames_valid() (*in module scilpy.io.utils*), 15
assert_inputs_exist() (*in module scilpy.io.utils*), 15
assert_output_dirs_exist_and_empty() (*in module scilpy.io.utils*), 15
assert_outputs_exist() (*in module scilpy.io.utils*), 15
assert_same_resolution() (*in module scilpy.io.image*), 13

B

B0ExtractionStrategy (*class in scilpy.utils.bvec_bval_tools*), 39
base_10_log() (*in module scilpy.image.operations*), 8
binary_classification() (*in module scilpy.tractanalysis.reproducibility_measures*), 31
build_ms_from_shell_idx() (*in module scilpy.viz.gradient_sampling*), 49

C

ceil() (*in module scilpy.image.operations*), 8
check_b0_threshold() (*in module scilpy.utils.bvec_bval_tools*), 39
check_slice_indices() (*in module scilpy.image.utils*), 12
check_tracts_same_format() (*in module scilpy.io.streamlines*), 13
check_tracts_same_format() (*in module scilpy.io.utils*), 16
closing() (*in module scilpy.image.operations*), 8
compress_sft() (*in module scilpy.utils.streamlines*), 47
compute_average_dir() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 34
compute_bundle_adjacency_streamlines() (*in module scilpy.tractanalysis.reproducibility_measures*), 32

compute_bundle_adjacency_voxel() (*in module scilpy.tractanalysis.reproducibility_measures*), 32
compute_bvalue_lin_b() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 5
compute_bvalue_lin_q() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 5
compute_connectivity() (*in module scilpy.tractanalysis.tools*), 38
compute_correlation() (*in module scilpy.tractanalysis.reproducibility_measures*), 32
compute_dice_streamlines() (*in module scilpy.tractanalysis.reproducibility_measures*), 32
compute_dice_voxel() (*in module scilpy.tractanalysis.reproducibility_measures*), 33
compute_distance_barycenters() (*in module scilpy.utils.util*), 48
compute_distance_to_peak() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 35
compute_dwi_attenuation() (*in module scilpy.reconst.raw_signal*), 22
compute_fodf() (*in module scilpy.reconst.fodf*), 19
compute_fractal_dimension() (*in module scilpy.tractanalysis.reproducibility_measures*), 33
compute_ks_from_shell_idx() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 5
compute_lesion_stats() (*in module scilpy.utils.metrics_tools*), 44
compute_min_duty_cycle_bruteforce() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 5
compute_msmt_frf() (*in module scilpy.reconst.frf*), 20
compute_peak_power() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 6
compute_sh_coefficients() (*in module scilpy.reconst.raw_signal*), 22
compute_snr() (*in module scilpy.utils.image*), 43
compute_ssst_frf() (*in module scilpy.reconst.frf*), 21
compute_streamline_segment() (*in module scilpy.tractanalysis.tools*), 38
compute_todi() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 35
compute_tract_counts_map() (*in module scilpy.tractanalysis.streamlines_metrics*), 34
compute_vectors_norm() (*in module scilpy.tractanalysis.todi_util*), 37
compute_weights() (*in module scilpy.gradientsampling.multiple_shell_energy*), 1
concatenate() (*in module scilpy.image.operations*), 8
convert() (*in module scilpy.image.operations*), 9
correct_b0s_philips() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 6
cost() (*in module scilpy.gradientsampling.multiple_shell_energy*), 2
count_non_zero_voxels() (*in module scilpy.image.utils*), 12
create_acqparams() (*in module scilpy.preprocessing.distortion_correction*), 17
create_index() (*in module scilpy.preprocessing.distortion_correction*), 18
create_multi_topup_index() (*in module scilpy.preprocessing.distortion_correction*), 18
create_non_zero_norm_bvecs() (*in module scilpy.preprocessing.distortion_correction*), 18
cut_between_masks_streamlines() (*in module scilpy.tractanalysis.tools*), 38
cut_invalid_streamlines() (*in module scilpy.utils.streamlines*), 47
cut_outside_of_mask_streamlines() (*in module scilpy.tractanalysis.tools*), 38

D

detect_ushape() (*in module scilpy.tractanalysis.features*), 29
difference() (*in module scilpy.image.operations*), 9
dilation() (*in module scilpy.image.operations*), 9
display_slices() (*in module scilpy.viz.screenshot*), 50
division() (*in module scilpy.image.operations*), 9

E

electrostatic_repulsion() (*in module scilpy.gradientsampling.multiple_shell_energy*), 2
equality_constraints() (*in module scilpy.gradientsampling.multiple_shell_energy*), 2
erosion() (*in module scilpy.image.operations*), 9
extract_affine() (*in module scilpy.image.utils*), 12
extract_b0() (*in module scilpy.utils.bvec_bval_tools*), 40

extract_dwi_shell() (*in module scilpy.utils.bvec_bval_tools*), 40
extract_longest_segments_from_profile() (*in module scilpy.tractanalysis.tools*), 38

F

filter_cuboid() (*in module scilpy.segment.streamlines*), 25
filter_ellipsoid() (*in module scilpy.segment.streamlines*), 25
filter_grid_roi() (*in module scilpy.segment.streamlines*), 25
filter_grid_roi_both() (*in module scilpy.segment.streamlines*), 26
filter_streamlines_by_length() (*in module scilpy.tracking.tools*), 27
filter_streamlines_by_total_length_per_dim() (*in module scilpy.tracking.tools*), 28
filter_tractogram_data() (*in module scilpy.utils.streamlines*), 47
find_order_from_nb_coeff() (*in module scilpy.reconst.utils*), 23
FIRST (*scilpy.utils.bvec_bval_tools.B0ExtractionStrategy attribute*), 39
flip_fsl_gradient_sampling() (*in module scilpy.utils.bvec_bval_tools*), 41
flip_mrtrix_gradient_sampling() (*in module scilpy.utils.bvec_bval_tools*), 41
floor() (*in module scilpy.image.operations*), 9
fsl2mrtrix() (*in module scilpy.utils.bvec_bval_tools*), 41

G

gaussian_blur() (*in module scilpy.image.operations*), 9
generate_gradient_sampling() (*in module scilpy.gradientsampling.gen_gradient_sampling*), 1
generate_mask_indices_1d() (*in module scilpy.tractanalysis.todi_util*), 37
get_array_ops() (*in module scilpy.image.operations*), 9
get_b_matrix() (*in module scilpy.reconst.utils*), 23
get_bundle_metrics_mean_std() (*in module scilpy.utils.metrics_tools*), 44
get_bundle_metrics_mean_std_per_point() (*in module scilpy.utils.metrics_tools*), 45
get_bundle_metrics_profiles() (*in module scilpy.utils.metrics_tools*), 45
get_color_streamlines_along_length() (*in module scilpy.utils.streamlines*), 47
get_data_as_mask() (*in module scilpy.io.image*), 13
get_dir_to_sphere_id() (*in module scilpy.tractanalysis.todi_util*), 37
get_endpoints_density_map() (*in module scilpy.tractanalysis.reproducibility_measures*), 33
get_final_pruned_indices() (*scilpy.segment.recobundlesx.RecobundlesX method*), 24
get_head_tail_density_maps() (*in module scilpy.tractanalysis.reproducibility_measures*), 34
get_image_ops() (*in module scilpy.image.operations*), 9
get_indices_1d() (*in module scilpy.tractanalysis.todi_util*), 37
get_mask() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 35
get_maximas() (*in module scilpy.reconst.utils*), 23
get_next_real_point() (*in module scilpy.tractanalysis.quick_tools*), 31
get_operations_doc() (*in module scilpy.image.operations*), 9
get_point_on_line() (*in module scilpy.tractanalysis.tools*), 38
get_previous_real_point() (*in module scilpy.tractanalysis.quick_tools*), 31
get_roi_metrics_mean_std() (*in module scilpy.utils.metrics_tools*), 46
get_segments_dir_and_norm() (*in module scilpy.tractanalysis.todi_util*), 37
get_segments_mid_pts_positions() (*in module scilpy.tractanalysis.todi_util*), 37
get_segments_vectors() (*in module scilpy.tractanalysis.todi_util*), 37
get_sh() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 35
get_sh_order_and_fullness() (*in module scilpy.reconst.utils*), 23
get_shell_indices() (*in module scilpy.utils.bvec_bval_tools*), 41
get_sphere_neighbours() (*in module scilpy.reconst.utils*), 23
get_streamline_pt_index() (*in module scilpy.tractanalysis.tools*), 38
get_streamlines_bounding_box() (*in module scilpy.tractanalysis.features*), 29
get_streamlines_centroid() (*in module scilpy.tractanalysis.features*), 29
get_tdi() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
get_theta() (*in module scilpy.tracking.tools*), 28
get_todi() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
get_todi_shape() (*scilpy.tractanalysis.todi.TrackOrientationDensityI*

method), 36
get_vectors_dir_and_norm() (*in module scilpy.tractanalysis.todi_util*), 37
get_vectors_dir_and_norm_rel_to_center() (*in module scilpy.tractanalysis.todi_util*), 37
grad_cost() (*in module scilpy.gradientsampling.multiple_shell_energy*), 2
grad_electrostatic_repulsion() (*in module scilpy.gradientsampling.multiple_shell_energy*), 3
grad_equality_constraints() (*in module scilpy.gradientsampling.multiple_shell_energy*), 3
grid_intersections() (*in module scilpy.tractanalysis.grid_intersections*), 31

I

ichunk() (*in module scilpy.io.streamlines*), 13
identify_shells() (*in module scilpy.utils.bvec_bval_tools*), 41
intersection() (*in module scilpy.image.operations*), 9
intersects_two_rois() (*in module scilpy.tractanalysis.tools*), 39
invert() (*in module scilpy.image.operations*), 9
is_argument_set() (*in module scilpy.io.streamlines*), 13
is_float() (*in module scilpy.utils.util*), 48
is_header_compatible_multiple_files() (*in module scilpy.io.utils*), 16
is_normalized_bvecs() (*in module scilpy.utils.bvec_bval_tools*), 42

L

link_bundles_and_reference() (*in module scilpy.io.utils*), 16
load_matrix_in_any_format() (*in module scilpy.io.utils*), 16
load_tractogram_with_reference() (*in module scilpy.io.streamlines*), 13
lower_clip() (*in module scilpy.image.operations*), 9
lower_threshold() (*in module scilpy.image.operations*), 9
lower_threshold_eq() (*in module scilpy.image.operations*), 9

M

mask_todi() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
MEAN (*scilpy.utils.bvec_bval_tools.B0ExtractionStrategy attribute*), 39
mean() (*in module scilpy.image.operations*), 9
merge_labels_into_mask() (*in module scilpy.io.image*), 13
min_dist_to_centroid() (*in module scilpy.tractanalysis.distance_to_centroid*), 29
mrtrix2fsl() (*in module scilpy.utils.bvec_bval_tools*), 42
multiple_shell() (*in module scilpy.gradientsampling.multiple_shell_energy*), 3
multiplication() (*in module scilpy.image.operations*), 10

N

natural_log() (*in module scilpy.image.operations*), 10
normalize_bvecs() (*in module scilpy.utils.bvec_bval_tools*), 42
normalize_max() (*in module scilpy.image.operations*), 10
normalize_sum() (*in module scilpy.image.operations*), 10
normalize_todi_per_voxel() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
normalize_vectors() (*in module scilpy.tractanalysis.todi_util*), 37

O

opening() (*in module scilpy.image.operations*), 10
outliers_removal_using_hierarchical_quickbundles() (*in module scilpy.tractanalysis.features*), 29

P

p_normalize_vectors() (*in module scilpy.tractanalysis.todi_util*), 37
parser_color_type() (*in module scilpy.io.utils*), 16
plot_each_shell() (*in module scilpy.viz.gradient_sampling*), 49
plot_metrics_stats() (*in module scilpy.utils.metrics_tools*), 46
plot_proj_shell() (*in module scilpy.viz.gradient_sampling*), 49
pre_filtering_for_geometrical_shape() (*in module scilpy.segment.streamlines*), 26
prune() (*in module scilpy.tractanalysis.features*), 30
prune_far_from_model() (*scilpy.segment.recobundlesx.RecobundlesX method*), 24
psf_from_sphere() (*in module scilpy.tractanalysis.todi_util*), 37

R

random_uniform_on_sphere() (*in module scilpy.gradientsampling.multiple_shell_energy*), 3
ranged_type() (*in module scilpy.io.utils*), 16
read_info_from_mb_bdo() (*in module scilpy.io.utils*), 16
RecobundlesX (*class in scilpy.segment.recobundlesx*), 24
recognize() (*scilpy.segment.recobundlesx.RecobundlesX method*), 24
reconstruct_streamlines() (*in module scilpy.io.streamlines*), 14
reconstruct_streamlines_from_hdf5() (*in module scilpy.io.streamlines*), 14
reconstruct_streamlines_from_memmap() (*in module scilpy.io.streamlines*), 14
register_image() (*in module scilpy.utils.image*), 43
remove_loops_and_sharp_turns() (*in module scilpy.tractanalysis.features*), 30
remove_outliers() (*in module scilpy.tractanalysis.features*), 30
remove_similar_streamlines() (*in module scilpy.segment.models*), 24
resample_streamlines_num_points() (*in module scilpy.tracking.tools*), 28
resample_streamlines_step_size() (*in module scilpy.tracking.tools*), 28
resample_volume() (*in module scilpy.image.resample_volume*), 10
reshape_to_3d() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
reslice() (*in module scilpy.image.reslice*), 11

S

sample_distribution() (*in module scilpy.tracking.tools*), 29
save_gradient_sampling_fsl() (*in module scilpy.gradientsampling.save_gradient_sampling*), 7
save_gradient_sampling_mrtrix() (*in module scilpy.gradientsampling.save_gradient_sampling*), 8
save_matrix_in_any_format() (*in module scilpy.io.utils*), 16
scilpy.gradientsampling.gen_gradient_sampling (*module*), 1
scilpy.gradientsampling.multiple_shell_energy (*module*), 1
scilpy.gradientsampling.optimize_gradient_sampling (*module*), 4
scilpy.gradientsampling.save_gradient_sampling (*module*), 7
scilpy.image.operations (*module*), 8
scilpy.image.resample_volume (*module*), 10
scilpy.image.reslice (*module*), 11
scilpy.image.utils (*module*), 12
scilpy.io.image (*module*), 13
scilpy.io.streamlines (*module*), 13
scilpy.io.utils (*module*), 14
scilpy.preprocessing.distortion_correction (*module*), 17
scilpy.reconst.afd_along_streamlines (*module*), 19
scilpy.reconst.fodf (*module*), 19
scilpy.reconst.frf (*module*), 20
scilpy.reconst.raw_signal (*module*), 22
scilpy.reconst.utils (*module*), 23
scilpy.segment.models (*module*), 24
scilpy.segment.recobundlesx (*module*), 24
scilpy.segment.streamlines (*module*), 25
scilpy.segment.voting_scheme (*module*), 26
scilpy.tracking.tools (*module*), 27
scilpy.tractanalysis.distance_to_centroid (*module*), 29
scilpy.tractanalysis.features (*module*), 29
scilpy.tractanalysis.grid_intersections (*module*), 31
scilpy.tractanalysis.quick_tools (*module*), 31
scilpy.tractanalysis.reproducibility_measures (*module*), 31
scilpy.tractanalysis.streamlines_metrics (*module*), 34
scilpy.tractanalysis.todi (*module*), 34
scilpy.tractanalysis.todi_util (*module*), 37
scilpy.tractanalysis.tools (*module*), 38
scilpy.tractanalysis.uncompress (*module*), 39
scilpy.utils.bvec_bval_tools (*module*), 39
scilpy.utils.filenames (*module*), 43
scilpy.utils.image (*module*), 43
scilpy.utils.metrics_tools (*module*), 44
scilpy.utils.streamlines (*module*), 47
scilpy.utils.util (*module*), 48
scilpy.viz.gradient_sampling (*module*), 49
scilpy.viz.screenshot (*module*), 50
set_todi() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
single_clusterize_and_rbx_init() (*in module scilpy.segment.voting_scheme*), 26
single_recognize() (*in module scilpy.segment.voting_scheme*), 27
smooth_line_gaussian() (*in module scilpy.tracking.tools*), 29
smooth_line_spline() (*in module scilpy.tracking.tools*), 29

smooth_todi_dir() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
smooth_todi_spatial() (*scilpy.tractanalysis.todi.TrackOrientationDensityImaging method*), 36
snapshot() (*in module scilpy.io.utils*), 16
split_heads_tails_kmeans() (*in module scilpy.tractanalysis.tools*), 39
split_name_with_nii() (*in module scilpy.utils.filenames*), 43
std() (*in module scilpy.image.operations*), 10
str_to_index() (*in module scilpy.utils.util*), 48
streamlines_in_mask() (*in module scilpy.segment.streamlines*), 26
streamlines_to_endpoints() (*in module scilpy.tractanalysis.todi_util*), 37
streamlines_to_memmap() (*in module scilpy.io.streamlines*), 14
streamlines_to_pts_dir_norm() (*in module scilpy.tractanalysis.todi_util*), 37
streamlines_to_segments() (*in module scilpy.tractanalysis.todi_util*), 38
subtraction() (*in module scilpy.image.operations*), 10
swap_fsl_gradient_axis() (*in module scilpy.utils.bvec_bval_tools*), 42
swap_mrtrix_gradient_axis() (*in module scilpy.utils.bvec_bval_tools*), 42
swap_sampling_eddy() (*in module scilpy.gradientsampling.optimize_gradient_sampling*), 7

T

TrackOrientationDensityImaging (*class in scilpy.tractanalysis.todi*), 34
transform_anatomy() (*in module scilpy.utils.image*), 43
transform_dwi() (*in module scilpy.utils.image*), 44

U

uncompress() (*in module scilpy.tractanalysis.uncompress*), 39
uniformize_bundle_sft() (*in module scilpy.utils.streamlines*), 48
uniformize_bundle_sft_using_mask() (*in module scilpy.utils.streamlines*), 48
union() (*in module scilpy.image.operations*), 10
upper_clip() (*in module scilpy.image.operations*), 10
upper_threshold() (*in module scilpy.image.operations*), 10
upper_threshold_eq() (*in module scilpy.image.operations*), 10

V

validate_nbr_processes() (*in module scilpy.io.utils*), 17
validate_sh_basis_choice() (*in module scilpy.io.utils*), 17
verify_compatibility_with_reference_sft() (*in module scilpy.io.utils*), 17
verify_compression_th() (*in module scilpy.io.utils*), 17
volume_iterator() (*in module scilpy.image.utils*), 12
VotingScheme (*class in scilpy.segment.voting_scheme*), 26
voxel_to_world() (*in module scilpy.utils.util*), 48

W

weighted_mean_std() (*in module scilpy.utils.metrics_tools*), 46
world_to_voxel() (*in module scilpy.utils.util*), 49
write_multiple_shells() (*in module scilpy.gradientsampling.multiple_shell_energy*), 4