Population-specific tractography bundle atlas creation

GSoC 2021 INCF - DIPY

Student
David Romero-Bascones

Mentors
Bramsh Qamar Chandio
Shreyas Fadnavis
Jong Sung Park

August 17, 2021

Abstract

This report describes the work done during Google Summer of Code 2021 within DIPY community [1], which is part of INCF organization. The main goal of the project is to develop a tool to build population specific atlases of white matter bundles in streamline space. The following sections detail the developed implementation as well as the research results related with bundle registration, bundle combination and atlasing.

Contents

1 Approach

The main goal of this project is to develop a tool for population-specific bundle atlas building. Given a set of segmented bundles as input, the tool should be able to combine them into a single bundle that captures well the population anatomy (Fig. 1).

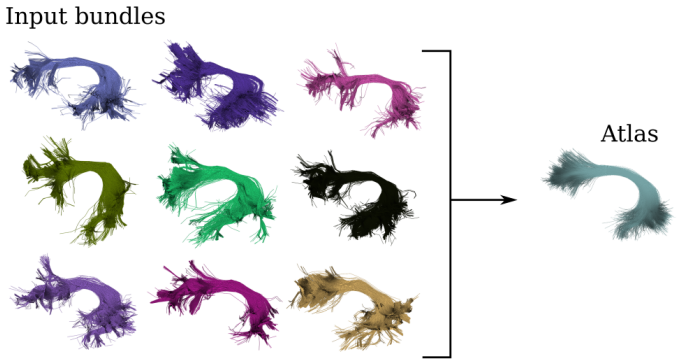

Figure 1: Population bundle atlas creation

Building a bundle atlas in streamline space posses two main challenges:

- **Construct a common space:** when described by the originally acquired coordinates bundles are said to be in native space. This representation reflects accurately the anatomy of each subject but does not allow to combine information across subjects as each bundle/subject has a specific coordinate system. One of the goals of atlasing is to build a common space that is shared between all bundles.

  For this, the common approach would be to select a bundle and register all the others to that space. This, although simple, generates a space that is highly biased towards that subject, which is not desirable. As an alternative, in this project we have followed a pairwise-matching approach described in Fig. 2. Bundles are registered and combined in pairs until a single (and final) result is obtained. This helps to disregard the influence of one specific bundle and obtain a less biased result.

  In addition, alternatives to common streamline linear registration were explored (see section 2.2).

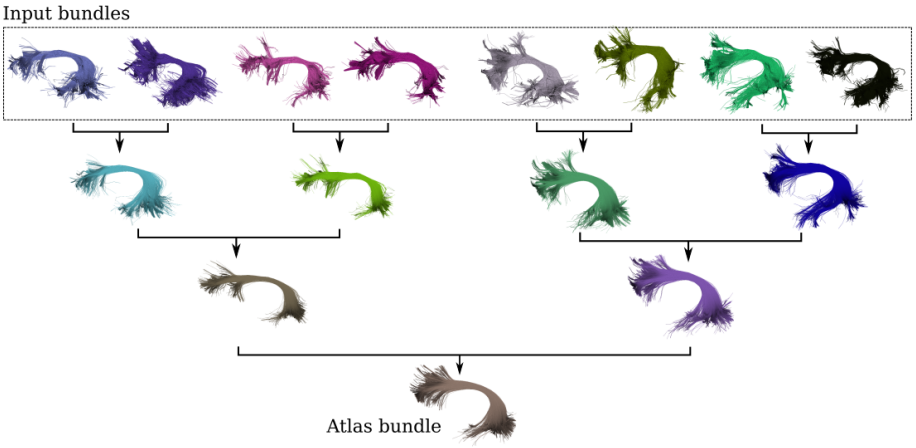

Figure 2: Pairwise tree atlas building.

- **Combine the bundles:** after registering a pair of bundles to a common space it is not obvious how these should be combined. In principle, streamlines can be either kept as they are or averaged between the two bundles. Here multiple bundle combination methods were tested (see section 2.3.1)

2 Research

2.1 Datasets

Two datasets were used for experiments:

- PPMI: used for benchmarking. It consists of 30 segmented bundles of 32 healthy controls and 32 patients from Parkinson Progression Markers Initiative (PPMI) data and used in [2].
- HCP-10: a second validation dataset derived from Human Connectome Project (HCP) data. It includes 72 segmented bundles from 10 healthy subjects used to validate TractSeg bundle segmentation algorithm [3].

In addition, the following datasets might be useful in the future:

- HCP-minor: Data from [4].
- Franco's data
- Infant-brain data from [5].

2.2 Bundle registration

2.2.1 Halfway Streamline Linear Regression (HSLR)

The common bundle registration approach aims to find the transformation matrix $\boldsymbol{T}_{12}$ that maps the $\boldsymbol{x}$ coordinates in bundle1 (moving) to the bundle2 space $\boldsymbol{x}'$ (static).

$$\boldsymbol{x}' = \boldsymbol{T}_{12}\boldsymbol{x} \tag{1}$$

Figure 3 shows the results of this approach obtained by streamline linear registration (SLR) [6].

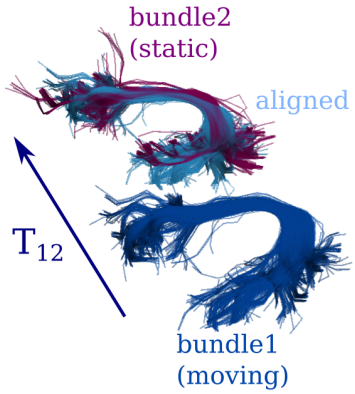

Figure 3: SLR of two bundles

Problematically, the results of this approach are highly biased by the bundle that is chosen as static. While this is desirable when registering bundles to an existing atlas, it is not optimal for atlas building. In fact, to obtain an atlas that is representative of the population it is important that the end result is not biased towards any specific subject.

Although using a tree-like atlas building might help reduce this bias, the end result will still be influenced by the arbitrary bundle that was chosen as static. To overcome that, halfway streamline linear registration (HSLR) aims to move both static and moving bundles to a common space that lies in the middle of them. The idea is rooted in the fact that the full transformation $\boldsymbol{T}_{12}$ can be split into two steps:

$$\boldsymbol{T}_{12} = \boldsymbol{T}_{m2}\boldsymbol{T}_{1m} \tag{2}$$

where $\boldsymbol{T}_{1m}$ is the transformation from bundle1 space to the middle space and $\boldsymbol{T}_{m2}$ is the transformation from the middle space to bundle2. To compute $\boldsymbol{T}_{1m}$ we first run SLR to obtain the full transformation $\boldsymbol{T}_{12}$. Then, $\boldsymbol{T}_{1m}$ is obtained

by dividing translation, rotation and shearing by half. Scaling is obtained as the mean between the value and 1:

$$\boldsymbol{T}_{1m} = \begin{cases} t'_x = t_x/2,\ t'_y = t_y/2,\ t'_z = t_z/2 \\ \theta'_x = \theta_x/2,\ \theta'_y = \theta_y/2,\ s'_z = \theta_z/2 \\ s'_x = (1+s_x)/2,\ s'_y = (1+s_y)/2,\ s'_z = (1+s_z)/2 \\ sh'_x = sh_x/2,\ sh'_y = sh_y/2,\ sh'_z = sh_z/2 \end{cases} \tag{3}$$

Now, by means of $\boldsymbol{T}_{1m}$, bundle1 is moved to the middle space (Fig. 4):

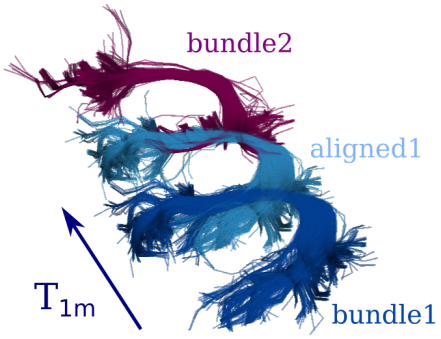

Figure 4: Step 1 of HSLR: move one bundle to the middle space.

The reciprocal matrix $\boldsymbol{T}_{2m}$ can be obtained by different strategies:

- **Analytical:** by operating over equation 2:

$$\boldsymbol{T}_{12}\boldsymbol{T}_{1m}^{-1} = \boldsymbol{T}_{m2}\boldsymbol{T}_{1m}\boldsymbol{T}_{1m}^{-1} \tag{4}$$

$$\boldsymbol{T}_{12}\boldsymbol{T}_{1m}^{-1} = \boldsymbol{T}_{m2} \tag{5}$$

$$\boldsymbol{T}_{2m} = \left(\boldsymbol{T}_{12}\boldsymbol{T}_{1m}^{-1}\right)^{-1} \tag{6}$$

- **Symmetric:** by following the same procedure used to compute $\boldsymbol{T}_{1m}$ but using bundle1 as static and bundle2 as moving.
- **Progressive:** by considering the bundle registered to the halfway space as the static bundle and then using SLR to align bundle2 to it.

With $\boldsymbol{T}_{2m}$ bundle2 is finally moved to the middle space and the registration procedure ends.

Figure 5: Step 2 of HSLR: move the second bundle to the middle space.

2.2.2 Method comparison

To test the validity of HSLR a total of 300 left arcuate fasciculus (AF) pairs from PPMI data were registered (affine) using the following approaches:

- SLR
- SLR inverted. The inverse of $\boldsymbol{T}_{12}$ was used to move the static bundle to the moving space ($\boldsymbol{T}_{21} = \boldsymbol{T}_{12}^{-1}$). This is to check symmetry and see if taking inverses has an impact on the accuracy.
- SLR progressive: we register bundle1 to bundle2 by moving it first to the middle space ($\boldsymbol{T}_{1m}$) and then running SLR again to move it from middle space to bundle2 space ($\boldsymbol{T}_{m2}$). This is to check if doing it in two steps improves the accuracy as with HSLR progressive.
- HSLR analytical
- HSLR progressive
- HSLR symmetric

methods were compared based on their registration accuracy, which was measured by bundle minimum distance (BMD) based on minimum direct flip (MDF) distance.

As expected, all methods managed to register bundles to a common space (Fig. 21). There were, however, important differences across methods (Fig. 6 and Table 2.2.2). Analytical approaches (SLR inverted and HSLR analytical) obtained worse results than regular SLR. This might indicate that computing inverses analytically has a negative impact on accuracy, including important outliers.

Running SLR twice did improve the accuracy only for the HSLR setting. More concretely, *HSLR progressive* improved SLR accuracy by 17.1%. These results point to **HSLR as a valid technique that is even able to boost SLR accuracy at the cost of running SLR twice**.

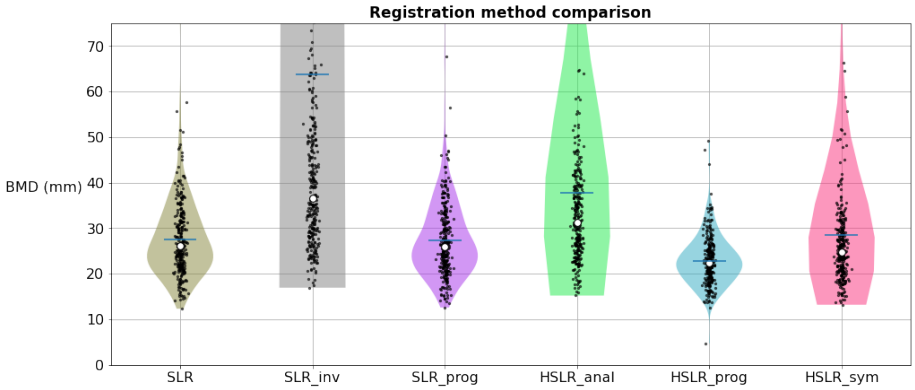

Figure 6: Comparison of registration methods accuracy.

Method	BMD (mm)	
	Mean ± SD	IQR
Native space	549.6 ± 600.1	[168.0, 717.5]
SLR	27.5 ± 8.2	[21.8, 31.7]
SLR inverted	63.7 ± 375.6	[29.0, 49.2]
SLR progressive	27.4 ± 8.4	[21.7, 32.1]
HSLR analytical	37.7 ± 74.0	[25.6, 38.9]
HSLR progressive	22.8 ± 5.4	[19.3, 25.5]
HSLR symmetric	28.6 ± 42.2	[20.8, 29.6]

Table 1: Results of registration method comparison.

2.2.3 SLR optimization symmetry

Related to the unbiased registration pursued by HSLR a question arises: if we run SLR considering considering bundle1 and bundle2 as static and moving and then, the other way around, do we obtain reciprocal transformation parameters?

To investigate the optimization symmetry of SLR a total of 200 left AF pairs were registered in both directions (switching static and moving) by using translation, rigid (translation + rotation), rigid + scaling and affine transformations. Linear correlation was measured by Spearman's r.

Results for translation (Fig. 7) show an almost perfect correlation. For rigid transform results were still highly symmetric (Fig. 8) with all correlations above 0.9. When introducing scaling into the transform the symmetry decreased notably. Surprisingly, scaling parameters did not follow clearly the expected $y = \frac{1}{x}$ symmetry. For instance, there were many cases where x scaling factor was bellow 1 for both directions, which means that the x dimension was squeezed no matter which bundle was considered as static.

Finally, the asymmetries observed for scaling were accentuated with affine transform, being shearing effect the most difficult to interpret.

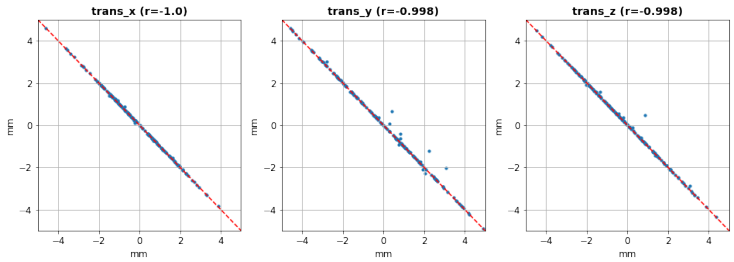

Figure 7: SLR symmetry for translation.

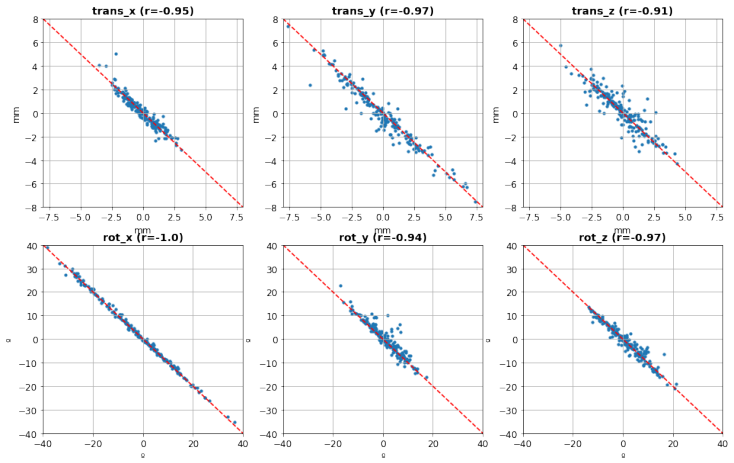

Figure 8: SLR symmetry for rigid transform.

An explanation for the scaling and affine asymmetries might be model complexity. As we allow the transformation to vary over more dimensions (parameters) the optimization function might become complex with several (and different) combinations of parameters reaching the same accuracy. Moreover, these results are likely to be highly influenced by the characteristic shape of the bundle (in this case the left AF), which might have a tendency to be different across subjects in some parameters but not in others.

2.3 Bundle combination

2.3.1 Important concepts

After moving a pair of bundles to the same space (i.e. after registration) the goal of bundle combination is to obtain a single bundle that aggregates the information present in the two input bundles.

An important point is that, in principle, it is desirable to obtain a bundle with a similar number of streamlines as the two combined bundles. In fact,

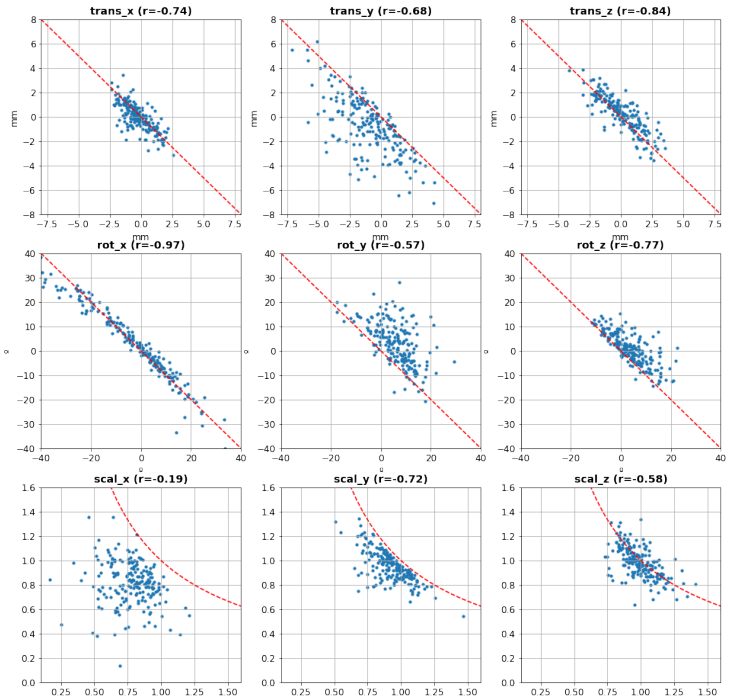

Figure 9: SLR symmetry for rigid + scaling transform.

keeping all the streamlines in both bundles without any discard would not scale well when combining more than one bundle. Therefore, one of the following bundle combination approaches are required:

- **Streamline selection:** a subset of the streamlines in both bundles are retained. The selection of streamlines can be done either randomly or based on any other metric.
- **Streamline combination:** streamlines from both bundles are first matched pairwise and then averaged. Streamline matching procedures rely on the calculation of distances between streamlines and therefore require a **distance metric**. Existing procedures range from simply finding the closest streamline to graph matching (GM) [7] or solving the linear assignment problem (LAP) [8]..

  An important issue arises when one bundle has less streamlines than the other. In those cases a 1 to 1 match is not possible and it needs to be decided what to do with the remaining streamlines, which can be:

  - Discard them
  - Keep them as they are

– Match them (1 to N matching) following one of the methods used to match streamlines.

2.3.2 Implemented methods

To illustrate every bundle combination method the pair of AF bundles shown in Fig. 10 was used.

Figure 10: Two test bundles with 4171 and 6038 streamlines, respectively.

The following **streamline-selection** methods were implemented:

- **Random-pick:** random selection of streamlines. Figure 11 shows the combination results with for different number of streamlines. Although computationally fast, results look noisy as streamlines are not being averaged.

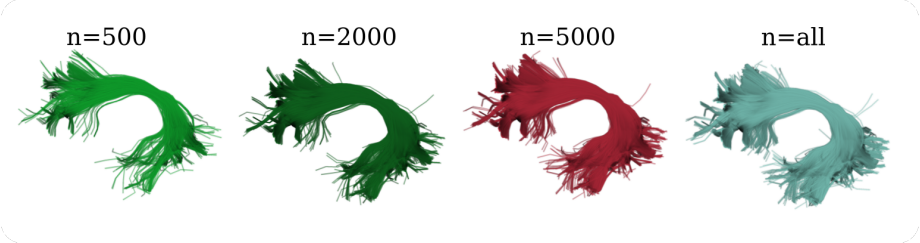

Figure 11: Bundle combination by random selection of streamlines

Regarding **streamline-combination** methods, two distance metrics were used: MDF and minimum direct flip start and endpoint (MDF-SE). The latter was meant to be a computationally fast that favors streamlines connecting the same regions. Using those distances the following bundle combination methods were implemented:

- **Closest-streamline:** streamlines are matched by looping through each streamline in bundle1 and finding the closest in bundle2. Two variants are distinguished (see Fig. 12):

– **closest-keep:** streamlines in bundle2 can be matched more than once to streamlines in bundle1. This results in a bundle with n_1 streamlines and a shape very close to the original bundle1. As streamlines can be matched more than once, it is likely that a few streamlines from bundle2 are being averaged. Therefore, this method is generally highly biased towards bundle1.

– **closet-remove:** streamlines are not allowed to be matched more than once. This reduces the bias towards bundle1 but is suboptimal. As we are looping through streamlines in bundle1 randomly (and discarding matched ones) it is unlikely that we are matching streamlines in an optimal way. In this sense, LAP offers a better solution.

Figure 12: Bundle combination by averaging the closest streamline

- **LAP:** in the general case of $n_1 < n_2$ this approach solves the so-called rectangular linear assignment problem (RLAP) by computing a 1-1 matching between streamlines in bundle1 and bundle2. The procedure minimizes the sum of pairwise distances (as opposed to closest-keep that minimizes the distance of each streamline pair individually). It was proposed in [8].

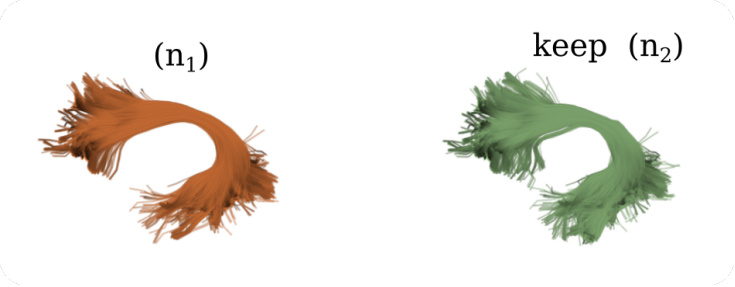

Figure 13: Bundle combination by LAP both removing unmatched streamlines and keeping them as they are (keep).

After the matching of n_1 streamlines, the remaining $n_2 - n_1$ can be either discarded or kept. Figure 13 shows the result of both approaches. As a

drawback, the computational cost starts to become a burden when the number of streamlines is > 2000.

- **GM:** proposed in [7], it aims to match streamlines based on their relationship with other streamlines within the same bundle. The idea is that the adjacency matrix that defines its structure is consistent across subjects. It requires both bundles to have the same number of streamlines. When the number of streamlines increases the computation can get really slow. Negatively, experiments with moving the same bundle show that GM is not able to match the streamlines perfectly.

Figure 14: Bundle combination by GM.

2.3.3 Robust streamline averaging

As noticed during the project, sometimes the matching is not perfect and averaging streamlines that are significantly distant might result in incorrect results. An example of this with HCP-10 data is shown in Fig. 15.

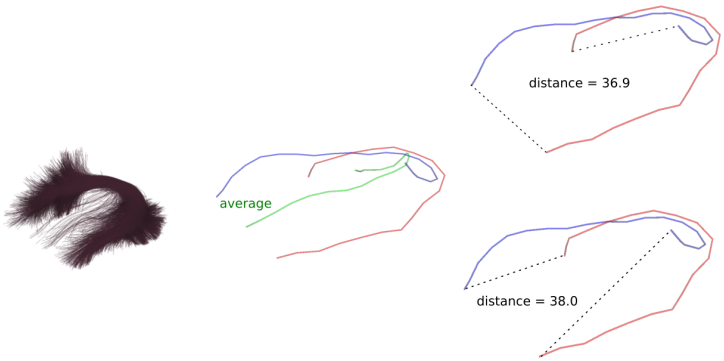

Figure 15: Spurious resulting from incorrect matching/averaging.

The reason for these spurious streamlines lies on several aspects:

- MDF-SE distance is smaller in the wrong orientation and does not reflect the actual orientation of streamlines. This is a very specific unlucky case

where the two matched streamlines have their start and endpoints very distant.

- All the streamlines in a bundle should be reoriented altogether (and not individually) prior to streamline matching.

To overcome this it is reasonable to not average and keep/discard those pairs with a distant higher than a certain threshold. A basic strategy could be:

1. Match streamlines and compute the distribution of pairwise distances
2. Translate distances to z-score
3. Do not match those with a z-score higher than some threshold (e.g. 3)
4. Decide what to do with unmatched streamlines (discard them or keep them as they are).

Fig. 16 shows how spurious streamlines disappear only when an outlier removal approach like this is applied. Reorienting streamlines correctly and using MDF instead of MDF-SE improves but does not totally correct the problem.

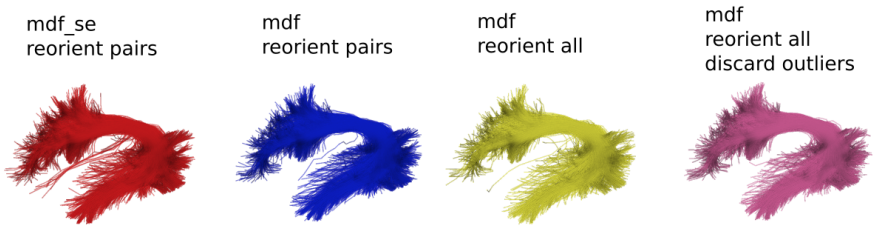

Figure 16: Spurious streamline correction.

2.4 Atlas building

Atlas benchmarking was thoroughly carried out for random_pick, rlap_keep and rlap_keep + skip pairs. The latter methods randomly skips averaging steps by using random_pick. This is intended to accelerate computation and diminish the smoothing consequence of averaging too much.

The closest-streamline methods were not fully tested as they provided highly biased results. Similarly, GM was computationally slow and did not provide better results than rlap.

First, the 32 controls of PPMI data were used to construct an atlas. Fig. 17 shows the resulting bundles from the three tested methods. Results from random_pick look rather noisy. On the contrary, the results from rlap_keep look significantly smoother.

As illustrated by Fig. 18 and 23, as the bundles are combined sharp details of bundles get smoothed. This is probably a consequence of the non-consistency

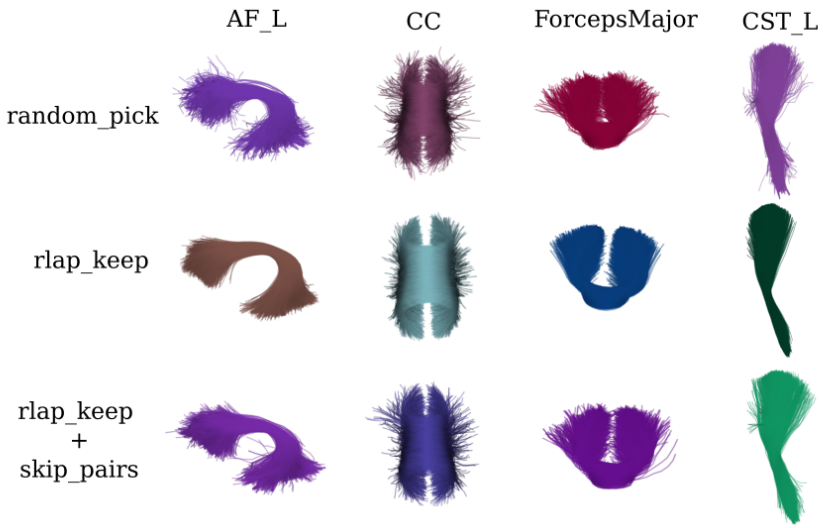

Figure 17: Atlas building results for PPMI dataset

of sharp features across subjects. For instance, the characteristic spike of the corticospinal tract (CST) is not present in every bundle so when a pair of bundles with and without it are averaged the spike progressively disappears.

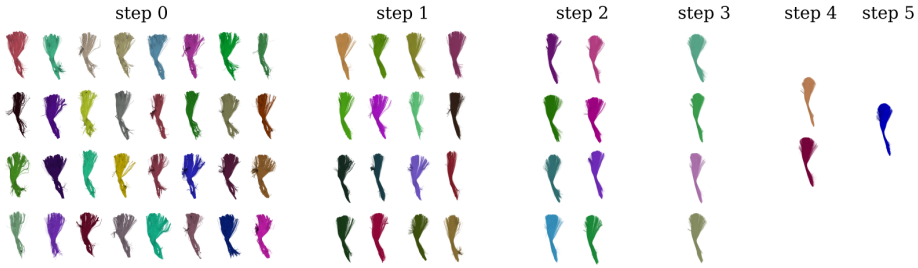

Figure 18: Atlas building of the leftCST tract

As a second validation the HCP-10 data was also used to construct an atlas. Results are shown in Fig. 19. Results look significantly different to those obtained for PPMI data. This probably reflects differences between the input bundles in both datasets, that were obtained by different techniques.

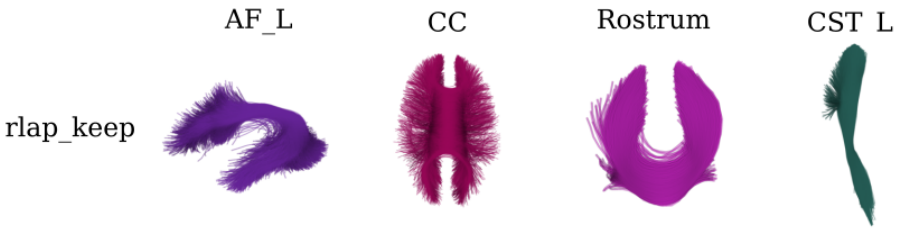

Figure 19: Atlas building results for HCP-10 dataset

2.5 Other ideas

2.5.1 Probabilistic atlas

Once the bundles are in the same space a probabilistic atlas can be constructed by calculating the percentage of subjects that have at least one streamline running through each voxel (coverage). An example of the resulting atlas is shown in Fig. 20 for the left AF.

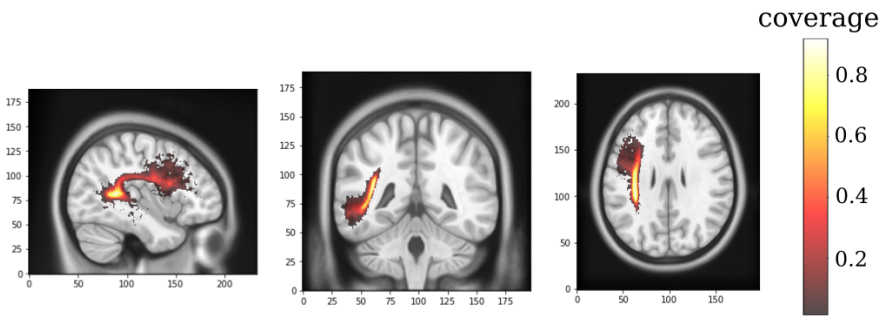

Figure 20: Probabilistic atlas of the left AF.

3 Implementation

The first implementation of the bundle atlas building workflow is based on a newly created **atlasing module**, which includes three main functions:

- get_pairwise_tree(): to generate a tree matching structure based on the number of initial items.
- combine_bundles(): combines a pair of bundles by using one of the following methods (rlap_keep or random_pick)

- compute_bundle_atlas(): implements the main workflow.

The workflow is also available as a command line interface (CLI) with the following arguments:

Argument	**Type**	**Description**
in_dir	str	Input folder with bundles to be processed
subjects	str	Subject list in a BIDS-like participants.tsv file
group	str	Group selector for participants.tsv
mid_path	str	Path between in_dir and bundle files
bundle_names	str	tsv file with bundles to be processed
model_bundle_dir	str	Folder with model bundles (atlas)
out_dir	str	Folder to save output
merge_out	Boolean	To merge all single bundles into a single file
save_temp	Bolean	To save intermediate steps in png and trk format
n_strea_min	Int	Bundles with less streamlines will be discarded
n_stream_max	Int	Maximum number of streamlines per bundle
n_point	Int	Points per streamline
distance	str	Distance metric to be used 'mdf' or 'mdf_se'
comb_method	str	Method used to combine bundle pairs
skip_pairs	Boolean	To randomly skip bundle combination steps

4 Future work

By topic of interest:

- HSLR
  - Test it with all streamlines and other bundles
  - Generate atlases using it
  - Eventually add it to the main workflow
- SLR symmetry. More comprehensive experiments:
  - Use all streamlines (and not only the centroids)
  - Test the repeatability of the optimization (i.e. do we get the same transformation when running SLR twice with the exact same configuration?)
  - Test it with other bundles (not just the left AF)
- Bundle combination
  - Add robust streamline averaging functionality to the main workflow
  - Try improving the speed of LAP by doing it hierarchically in clusters as proposed in the original work [8]

- Atlasing
  - Obtain new atlases with larger datasets. An interesting option might be infant-brain data.
  - Investigate further problematic bundles such as the vermis
  - Speed-up the process by using parallelization and cloud-computing (potential options: RAY, DASK or Cloudknot)
- Others
  - Add a probabilistic atlas workflow with group-wise statistical testing to DIPY

References

[1] E. Garyfallidis, M. Brett, B. Amirbekian, A. Rokem, S. van der Walt, M. Descoteaux, and I. N.-S. and, "Dipy, a library for the analysis of diffusion MRI data," *Frontiers in Neuroinformatics*, vol. 8, feb 2014.

[2] B. Q. Chandio, S. L. Risacher, F. Pestilli, D. Bullock, F.-C. Yeh, S. Koudoro, A. Rokem, J. Harezlak, and E. Garyfallidis, "Bundle analytics, a computational framework for investigating the shapes and profiles of brain pathways across populations," *Scientific Reports*, vol. 10, oct 2020.

[3] J. Wasserthal, P. Neher, and K. H. Maier-Hein, "TractSeg - fast and accurate white matter tract segmentation," *NeuroImage*, vol. 183, pp. 239–253, dec 2018.

[4] G. Bertò, D. Bullock, P. Astolfi, S. Hayashi, L. Zigiotto, L. Annicchiarico, F. Corsini, A. D. Benedictis, S. Sarubbo, F. Pestilli, P. Avesani, and E. Olivetti, "Classifyber, a robust streamline-based linear classifier for white matter bundle segmentation," *NeuroImage*, vol. 224, p. 117402, jan 2021.

[5] L. Walker, L.-C. Chang, A. Nayak, M. O. Irfanoglu, K. N. Botteron, J. McCracken, R. C. McKinstry, M. J. Rivkin, D.-J. Wang, J. Rumsey, and C. Pierpaoli, "The diffusion tensor imaging (DTI) component of the NIH MRI study of normal brain development (PedsDTI)," *NeuroImage*, vol. 124, pp. 1125–1130, jan 2016.

[6] E. Garyfallidis, O. Ocegueda, D. Wassermann, and M. Descoteaux, "Robust and efficient linear registration of white-matter fascicles in the space of streamlines," *NeuroImage*, vol. 117, pp. 124–140, aug 2015.

[7] E. Olivetti, N. Sharmin, and P. Avesani, "Alignment of tractograms as graph matching," *Frontiers in Neuroscience*, vol. 10, dec 2016.

[8] E. Olivetti, P. Gori, P. Astolfi, G. Bertó, and P. Avesani, "Nonlinear alignment of whole tractograms with the linear assignment problem," in *Biomedical Image Registration*, pp. 3–11, Springer International Publishing, 2020.

5 Appendix

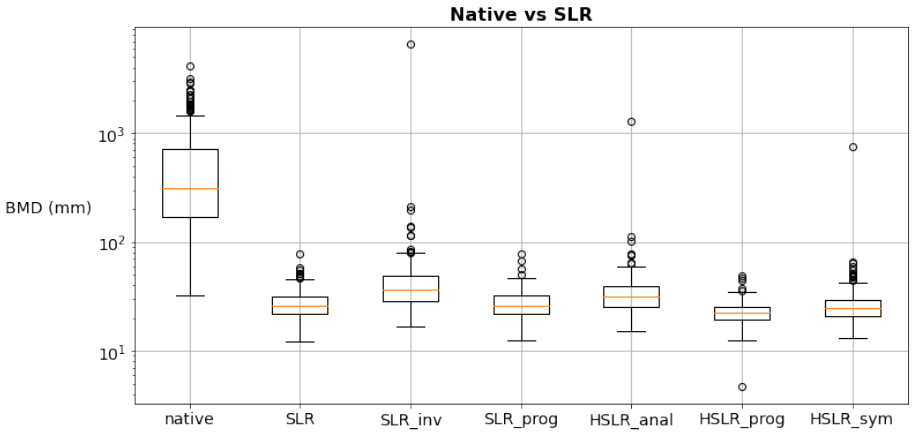

Figure 21: Accuracy of registration methods over native space.

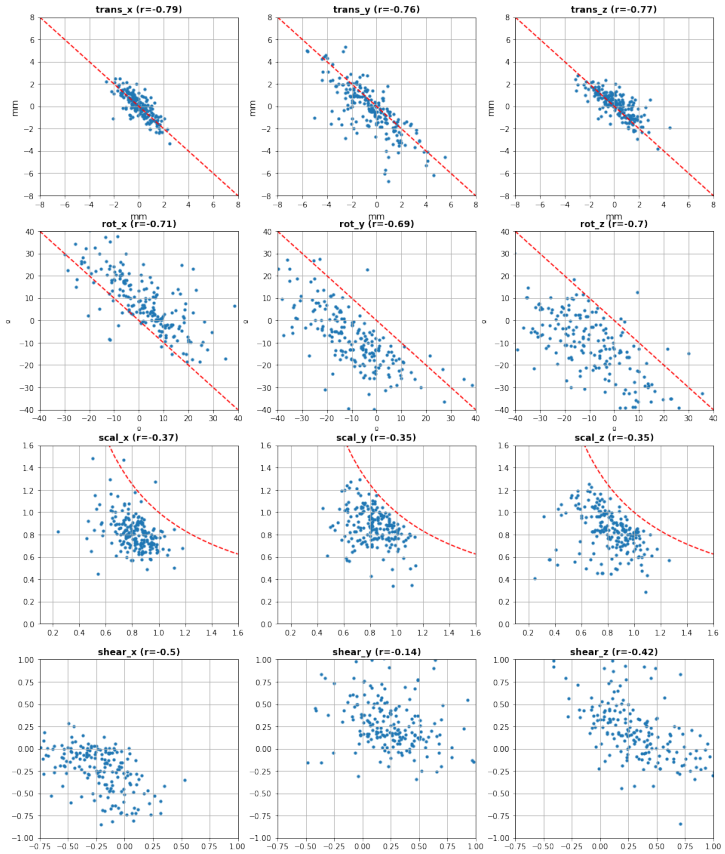

Figure 22: SLR symmetry for affine transform.

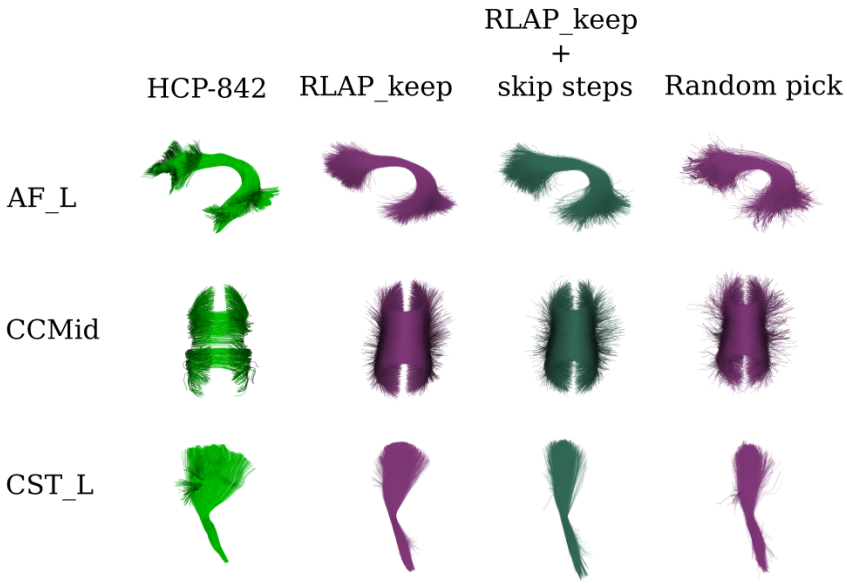

Figure 23: Atlas comparison with HCP