University of Verona

Department of Computer Science
Master's Degree in Medical Bioinformatics

Redundancy in tractography and its implications for the accuracy of brain connectivity estimates

Nicola Febbrari

VR451132

Supervisor:
Prof. Alessandro Daducci

Co-supervisor:
Dr. Simona Schiavi

Verona, 15/10/2021

Alle mie radici
mia sorella Gemma
i miei genitori

Abstract

Tractography is a family of algorithms that use *diffusion-weighted magnetic resonance imaging* data to reconstruct the white matter pathways of the brain non invasively. Despite its attractiveness, tractography reconstructions suffer from the presence of a large number of false positive pathways, or streamlines, which significantly bias any subsequent connectivity analysis based on this technique. Moreover, as a large number of possible streamlines are usually reconstructed, tractography reconstructions usually contain a lot of *redundancy*.

In this thesis, we focused our attention to the redundancy that is usually present in tractography reconstructions and, in particular, we *investigated its implications for the quality of brain connectivity estimates* using this technique. In fact, this redundancy among streamlines introduces collinearity problems in the microstructure-informed tractography methods that were recently proposed for mitigating the problem of false positives in the reconstructions, which may significantly reduce their performance.

To mitigate these collinearity problems, our group recently proposed a *new concept to represent streamlines*, called "blurred streamlines", which is based on a combination of clustering and spatial blurring of their signal contributions. *The aim of this thesis* was to systematically study the behavior of the blurred streamlines in order to better understand its strength and weaknesses. Quantitative analyses were conducted on synthetic data with known ground truth, focusing on the characterization on the accuracy and the repeatability of the method, as well as its computational cost.

Contents

1 Background **7**
1.1 Central nervous system 7
1.2 Magnetic Resonance Imaging 11
1.3 Tractography 17
1.4 Microstructure Informed Tractography 24

2 The problem of redundancy in tractography **30**
2.1 Redundancy and collinearity 30
2.2 Condition number 32
2.3 Removing redundancy using clustering 33

3 Clustering techniques **37**
3.1 Main clustering approaches 38
3.1.1 Centroid based 38
3.1.2 Hierarchical 42
3.1.3 Distribution based 43
3.1.4 Density based 43
3.1.5 Fuzzy 44
3.2 Clustering tractography data 45
3.2.1 QuickBundles 46
3.2.2 QuickBundlesX 50
3.2.3 FFClust 52

4 Experiments and Results **58**
4.1 Materials and methods 58

4.1.1 Data 58
4.1.2 Streamlines reconstruction 59
4.1.3 Clustering per bundle 61
4.1.4 Evaluation of the reconstructions 61
4.1.5 Libraries and software 62
4.2 Results 63
4.2.1 Collinearity and conditioning of the problem 63
4.2.2 Accuracy of the reconstructions 70
4.2.3 Repeatability 73

5 Conclusions and future perspectives **78**

Bibliography **81**

List of Figures

1.1 Composition of CNS 8
1.2 Composition of neuron 8
1.3 CSF 9
1.4 White and gray matter 10
1.5 Computers 11
1.6 Protons 12
1.7 Mrimachine 12
1.8 Tumor 13
1.9 Comparison MRI vs CT 13
1.10 Brownianmotion 14
1.11 T2-T1 16
1.12 Images dMRI with different b-values 16
1.13 DTI tractography 17
1.14 Example of tractography 18
1.15 Example tractogram used during the experiments 20
1.16 Deterministic tracking 20
1.17 Probabilistic tracking 21
1.18 Connectome 21
1.19 Invalid bundles 22
1.20 ROC curve 23
1.21 Ground-truth 23
1.22 Microstructure informed tractography concept 25
1.23 Approach top-down 26
1.24 Convex problem 27

1.25 Q-space model 28
1.26 COMMIT model 29

2.1 Collinearity in columns 31
2.2 SVD representation 33
2.3 Clustering tractography 34
2.4 Blur pipeline 34
2.5 Blurred effect in tractogram 36
2.6 Frenet-Serret frames 36

3.1 Unsupervised vs supervised learning 37
3.2 Different distances in machine learning 39
3.3 Elbow method 39
3.4 K-means example 41
3.5 Centroid approach 41
3.6 Hierarchical approach 42
3.7 Density approach 43
3.8 Density approach 44
3.9 Fuzzy approach 45
3.10 MDF distance 47
3.11 MDF bundle 48
3.12 CST bundle 49
3.13 Merge bundles 50
3.14 Benchmark QB 51
3.15 FFClust workflow 54
3.16 Thinner results 56

4.1 ISBI 3D 58
4.2 White matter 59
4.3 Bundle 44-45 61
4.4 Condition number test 65
4.5 Distance ground truth and COMMIT/COMMIT blur 67
4.6 Shifted points 68
4.7 Distance connectomes 69

4.8 Absolute difference connectomes 69
4.9 Valid bundles experiments 71
4.10 Invalid bundles experiments 72
4.11 RMSE experiments 72
4.12 Time experiments 73
4.13 RAM experiments 73
4.14 Repeatability definition 74
4.15 Valid bundles 50 runs 76
4.16 Invalid bundles 50 runs 77
4.17 RMSE 50 runs 77

Chapter 1

Background

1.1 Central nervous system

Human nervous system is a system that conducts stimuli from sensory receptors to the brain and spinal cord and conducts impulses back to other parts along the body [1]. It is divided into two main components: the central and the peripheral one. The central nervous system includes the brain and spinal cord, while the peripheral nervous system includes all of the nerves that branch out from the brain and spinal cord and extend to other parts of the body including muscles and organs [2].

The *central nervous system (CNS)* (Figure 1.1) controls most functions of the body and mind [3]. The brain is the microprocessor of our body, it plays a coordinating role. The brain is composed of neurons which are electrically excitable cells that communicate with other cells via specialized connections called synapses (Figure 1.2).

The *cerebrospinal fluid (CSF)* is a clear, colourless liquid that fills and surrounds the brain and the spinal cord and provides a mechanical barrier against shock [6]. Formed primarily in the ventricles of the brain, the cerebrospinal fluid supports the brain and provides lubrication between surrounding bones and the brain and spinal cord and this liquid plays a fundamental role in the self-regulation of cerebral blood flow (Figure 1.3).

Neurons are fundamental components in the "circuits" of nervous tissue.

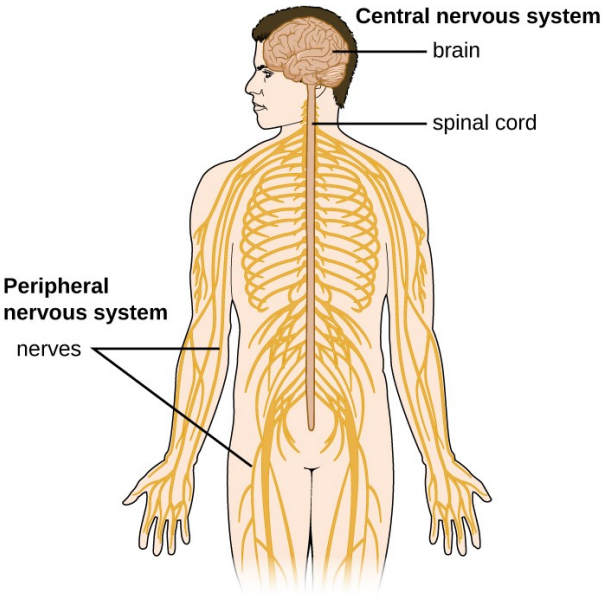


Figure 1.1: Representation of the human nervous system and its main components. Image taken from [4].

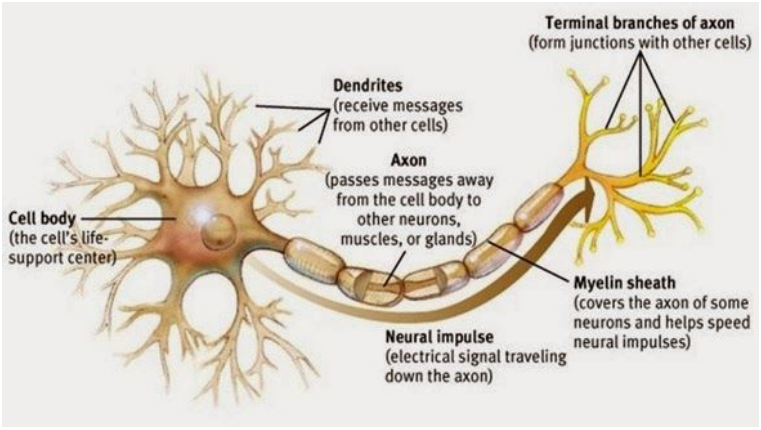


Figure 1.2: Representation of a neuron and its main components. Image taken from [5].

They generate variations in the electrical potential of the membrane to transfer information. These have a cell body (*soma*) that contains the nucleus and two types of extensions, respectively called *dendrites* and *axons*, along which the nerve impulse is transmitted. Dendrites receive the signal from neighboring neurons and propagate them in the centripetal direction. The *gray*

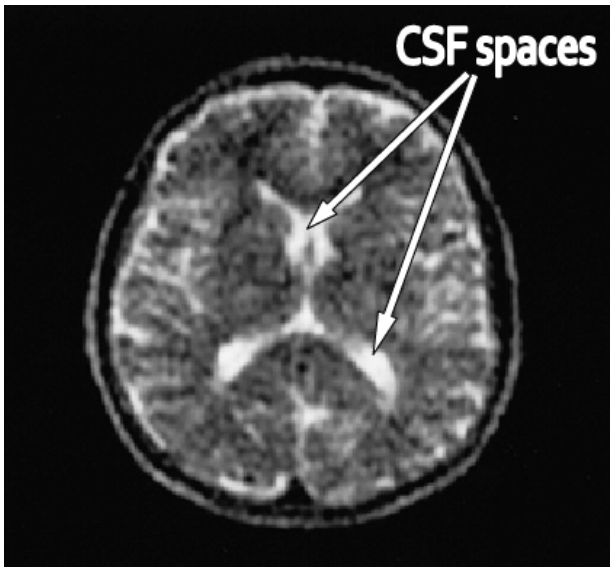


Figure 1.3: An example in which we can see CSF spaces. Image taken from [7].

matter is poorly myelinated, while in the *white matter* the concentration of myelin is very high, indeed the myelin increases the signal conduction, which allows to process and release information through the signaling of the axons in the white matter [8].

The *grey matter (GM)* is a major component of the central nervous system, consisting of neuronal cell bodies, neuropil (dendrites and unmyelinated axons), glial cells (astrocytes and oligodendrocytes), synapses and capillaries [9]. Grey matter is different from white matter because it contains numerous cell bodies and relatively few myelinated axons, while white matter contains relatively few cell bodies and is composed chiefly of long-range myelinated axons. The colour difference arises mainly from the whiteness of myelin. *GM* is found in the cortex (the most external area of the brain) and in some deeper areas, called deep gray matter or subcortical structures (Figure 1.4).

The grey matter performs the function of selection and initiation of information, it contains most of the brain's neuronal cell bodies. The grey matter includes regions of the brain involved in muscle control and sensory perception such as seeing and hearing, memory, emotions, speech, decision making, and self-control [11].

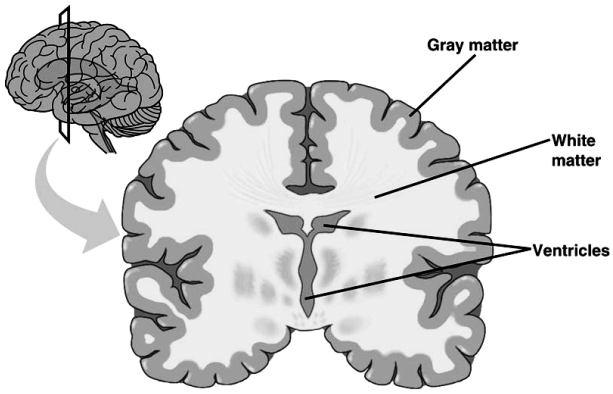


Figure 1.4: Representation of a brain where is highlighted the distinction between the gray and white matter. Image taken from [10].

The *white matter (WM)* refers to areas of the central nervous system (CNS) that are mainly made up of *myelinated axons* [9]. Axons can be myelinated or unmyelinated, and it is myelin's chemical composition of mainly lipids that gives the WM its characteristic color [12]. Myelin is a lipid-rich substance that surrounds nerve cell axons (the nervous system's "wires") to insulate them and increase the rate at which electrical impulses (called action potentials) are passed along the axon. The myelinated axon can be likened to an electrical wire (*axon*) with insulating material (*myelin*) around it. However, myelin does not form a single long sheath over the entire length of the axon. Rather, each myelin sheath insulates the axon over a single long section and, in general, each axon comprises multiple long myelinated sections separated from each other by short myelin sheath-gaps. Loss of myelin is a problem for many CNS disorders, including stroke, spinal cord injury, and multiple sclerosis (MS) [13]. Summarizing. the gray matter can be seen as the information processing area and signals, like a computer, while the white matter as the connections between the computers contained in a server room (Figure 1.5).

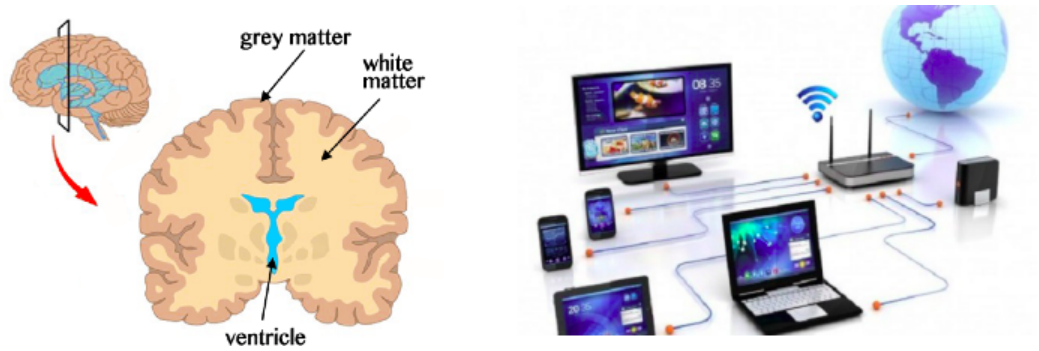


Figure 1.5: The analogy between the regions of the brain and the computer devices. Image taken from [14] [15].

1.2 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is a non-invasive imaging technology that produces three dimensional detailed anatomical images [16]. It is often employed for disease detection, treatment monitoring and diagnosis. It is based on sophisticated technology that detects the change in the direction of the rotational axis of protons found in the water that makes up living tissues [17]. MRIs consists of powerful magnets which produce a strong magnetic field that forces protons in the body to align with that field [18]. When a radiofrequency current is active in the patient, the protons are stimulated, and spin out of equilibrium, straining against the pull of the magnetic field [18]. When the radiofrequency field is inactive, the MRI sensors are able to detect the energy released as the protons realign with the magnetic field (Figure 1.6). To obtain an MRI image, a patient is placed inside a large magnet (Figure 1.7) and must stay very still during the imaging process in order not to blur the image. *The faster the protons realign, the brighter the image* [18].

The brain, spinal cord and nerves, as well as muscles, ligaments, and tendons are seen much more clearly with MRI than with regular x-rays and CT; for this reason MRI is often used to image knee and shoulder injuries. In the brain, MRI can differentiate between white matter and grey matter and can also be used to diagnose aneurysms and tumors (Figure 1.8). Because MRI does not use x-rays or other radiation, it is the imaging modality of

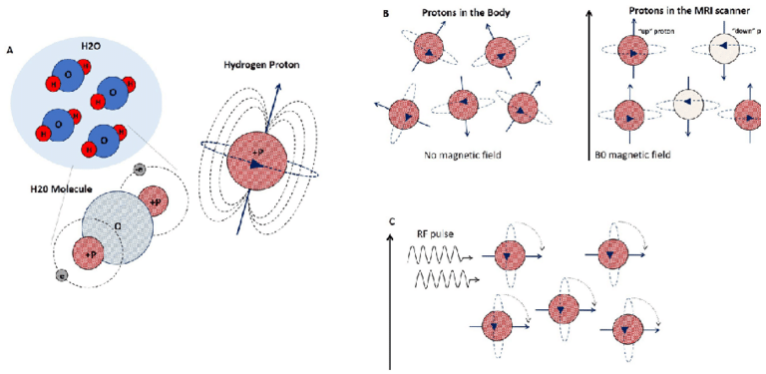


Figure 1.6: A representation of the protons behavior. Image taken from [19].

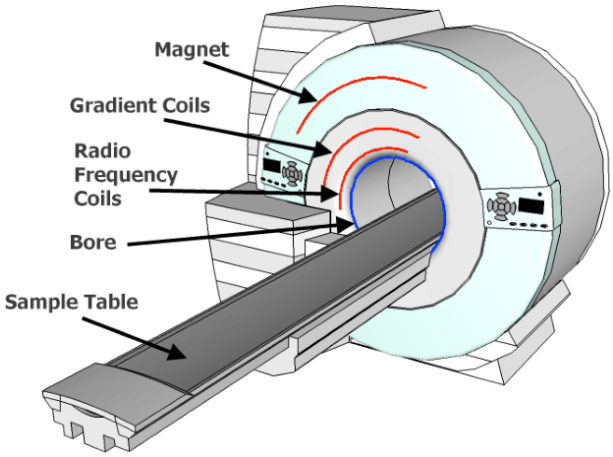


Figure 1.7: Diagram showing the MRI machine. Image taken from [20]

choice when frequent imaging is required for diagnosis or therapy, especially in the brain [21]. However, MRI is more expensive than x-ray imaging or CT scanning (Figure 1.9).

This thesis is based on information from *diffusion magnetic resonance images*, *dMRI*, which is used for the study of the connectivity of the brain based on the principle of diffusion (Brownian motion) (Figure 1.10).

According to this study, the molecules of a solution possess an incessant and disordered motion, which depends on: *diffusion time* and the *diffusion coefficient*, which in turn depends on the temperature, the type of molecule and the viscosity of the medium. Diffusion coefficients could be determined by measuring the concentration of different molecules at different times, using

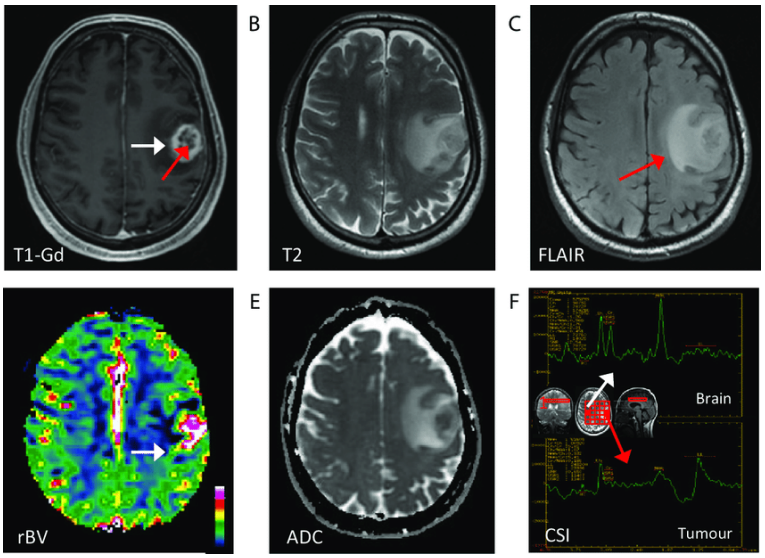


Figure 1.8: An example of the tumor showing thanks to MRI acquisition. Image taken from [22].

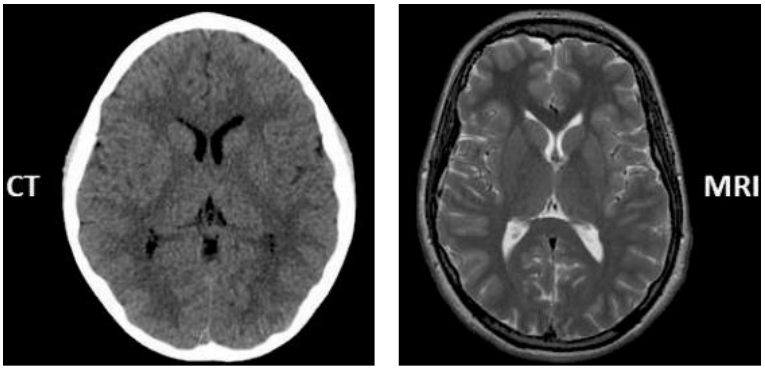


Figure 1.9: An example of the difference between MRI and CT. Image taken from [23].

physical or chemical methods based on classical Fick's first law [25]. In statistics, *diffusion is described as the mean squared path of molecules over a given time interval* (m^2/s). The equation that describes the diffusion is Einstein's equation of 1905:

$$< X >^2 = 2DT_d \tag{1.1}$$

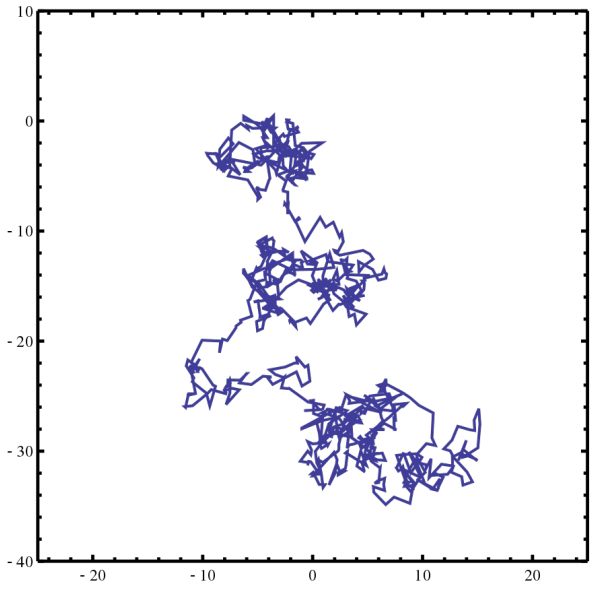


Figure 1.10: Diagram showing an example of the trajectory followed by a particle in Brownian motion. Image taken from [24].

where $< X >^2$ is the quadratic mean of the distance diffusion along one direction and T_d is the diffusion time [26]. **Diffusion MRI** focuses on the movement and directions of water molecules, particularly the spins of hydrogen ions, within the brain. *dMRI*, like *MRI*, is a *non-invasive* technique.

In the brain, the free and random movement of water is modified by the various *microstructural components* (sez 1.1). Einstein defined his equation for the random movement of water molecules in a glass; this cannot happen in human tissues that have a well-defined microstructure acting as an obstacle to the "free" movement. In diffusion, given the structural composition of the human brain, we focus on the movement and conformation of the axons and therefore on the white matter. Consequently, the coefficient used is not that of free [27]. To have diffusion images it is necessary to add a gradient to the magnetic field generated as for classical MRI images. In fact, in every MRI image the pulses of the gradient of the magnetic field are already present, but their amplitude and duration are usually too small to induce significant diffusion effects [27]. The hydrogen nucleus are immersed in the magnetic field, like gyroscopes immersed in a gravitational field. The *precession speed*,

i.e. the change of direction of the rotation axis, of the gyroscope depends on the gravitational force, just as the *precession speed of the hydrogen core depends on the intensity of the magnetic field in which absorbed*. If the external magnetic field is uniform, the precession rate will be the same for all nucleus and, consequently, the hydrogen atoms will all be in phase and the spin vector representing them will be given by the sum of all the single spins for that given voxel [28]. It is therefore necessary to insert in the field a linear gradient that varies with the speed of the single spins. This inserted gradient causes a phase shift that cannot be deleted by the application of this second gradient, since in the time interval between the application of the two magnetic fields the protons are definitively oriented. With the second gradient the spin shift is unchanged, the signal will not be the same as the previous gradient [28]. Consequently, the displacements of water molecules driven by diffusion are encoded in the MRI signal by spatial and temporal variation of the magnetic field generated by gradient pulses, given by the spins of the protons, in the magnetic field. The MRI signal, with these two gradient pulses, is attenuated and is called *"diffusion weighted"*.

It is important to quantify the effects of these gradients which attenuate the MRI signal. For this we use the *b-value* which is used to summarize all the effects of the gradient used. The *b-value* depends on the intensity of the diffusion gradients (q) and on the time interval of application of the two gradients (Δ) according to the following relationship:

$$b \propto q^2 * \Delta \tag{1.2}$$

Signal attenuation is then reduced to a simple expression:

$$signal \propto e^{-bD} \tag{1.3}$$

With *signal* we mean the magnetic resonance signal on the T2-weighted diffusion weighted image only, while D is the diffusion coefficient and b is the b-value (Figure 1.11).

The resulting raw images exhibit a certain degree of diffusion sensitiv-

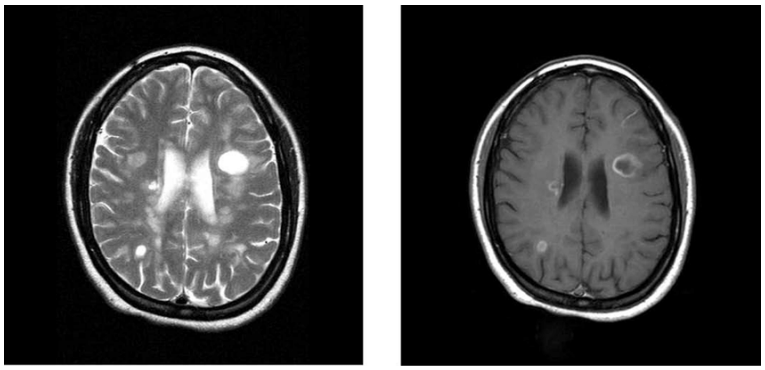


Figure 1.11: T2-weighted and gadolinium-enhanced T1-weighted magnetic resonance images demonstrating multiple sclerosis plaques. Image taken from [29].

ity, defined by the *b-value*. *The b-value is a factor that reflects the strength and timing of the gradients used to generate diffusion-weighted images.* The higher the *b-value*, the stronger the diffusion effects. These diffusion-weighted images are called *diffusion-weighted (DW)*. So the *color contrast of DW images* depends on: b-value, diffusion coefficient (D) and the gradient. In general: a higher b-value will give a darker image, this is because the signal of a particular tissue decreases exponentially as the b-value increases due to the molecular motion (Figure 1.12).

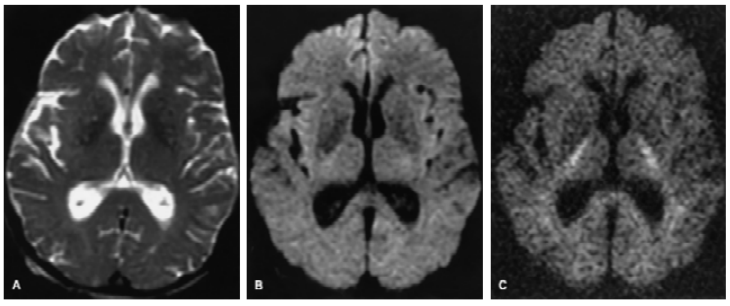


Figure 1.12: Some examples of dMRI with different b-value: $b = 0s/mm^2$, $b = 1000s/mm^2$ e $b = 3000s/mm^2$. Image taken from [30].

1.3 Tractography

Tractography, or *Fiber Tracking*, is a three-dimensional modeling technique used in neuroscience to visually represent neuronal traits reconstructed from data collected with dMRI. In this case, we have no single pixels, as two dimensional images, but *voxels*. *Tractography* is important for studying brain connectivity, hence the white matter bundles which are macroscopic structures that facilitate communication between brain regions at various distances. There may therefore be *short distance connections and long distance connections*. The short ones connect the adjacent convolutions (areas of the cerebral cortex bounded by two grooves) and are called *U fibers*, while the *connections between both hemispheres are called commissural fibers. The largest bundle of commissural fibers is the corpus callosum*, which connects the two hemispheres. *Projection fibers* connect the cerebral cortex with the lower part of the brain, such as the spinal cord (Figure 1.13).

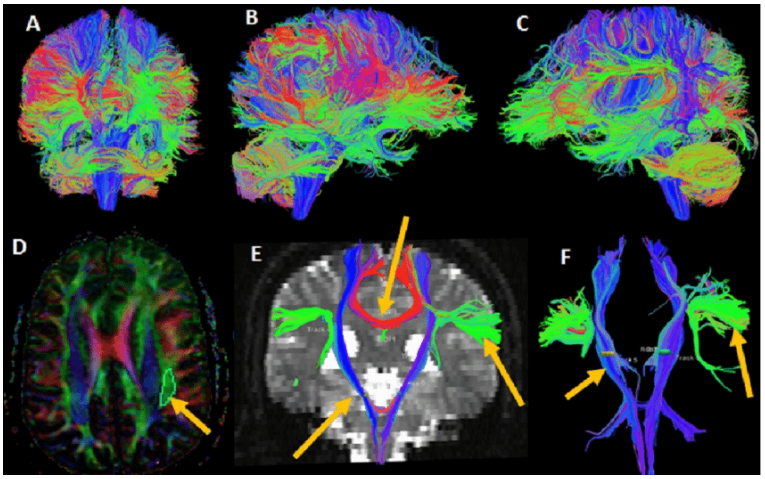


Figure 1.13: An example of DTI tractography. Image taken from [31].

The main idea on which the tractography is based and operates is to capture the coherence of the directions of the fibers by looking at the diffusion tensor [32] that describes each voxel. Tractography uses the measurement of *anisotropic diffusion* [33], there are various types of fibers and of different lengths, on a macroscopic scale and there are different tractographic algorithms to infer the macroscopic trajectories of the white matter. Using seeds

as starting points of the tractography algorithms, it is possible to generate polylines that represent and approximate groups of fibers in the brain's white matter. So we know that the diffusion tensor describes the displacement gradient in the three Cartesian axes and taking the most incisive direction it is possible to say that for that particular voxel there is that particular spatial inclination. Subsequently, the neighboring voxels are checked to see if there is an immediate continuation of the polyline, with respect to the same orientation, and consequently the voxel by voxel display is continued. Otherwise, it is necessary to go back, as the direction of the new voxel is completely different from the previous one (Figure 1.14).

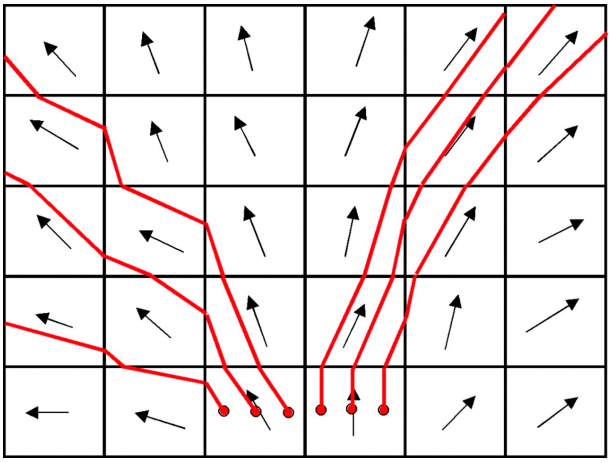


Figure 1.14: An example of basic algorithm for the creation of a streamline, in which the point x_0 is seen as the beginning. Image taken from [34].

Streamlines or polylines are therefore approximations of axons present in the human brain. These are estimated by integrating the partial differential equation:

$$\frac{\delta r(\tau)}{\delta \tau} = \nu_{traj}(r(\tau)) \tag{1.4}$$

where $r(\tau)$ is the path, δr is the temporal step and ν_{traj} is the vector field that defines the tangent of the local path direction. The simplest approach for creating the streamline path is to use Euler integration. Starting from the seeds or starting points, r_0 to obtain the orientation of the local fibers $\nu(r0)$, then following this direction for a short distance Δ, called sampling

step, in this way we can get the next point of the streamline. We can then reconstruct the path iteratively with this formula:

$$r_{i+1} = r_i + \nu(r_i)\Delta. \tag{1.5}$$

This method assumes that the $\nu(r_0)$ orientation is constant along the size of the Δ step. This makes the method highly sensitive in curved regions, as they may be skipped, especially for very long step sizes(Δ) [35]. The seeds can be placed on the whole brain or on some *regions of interest (ROI)*. Starting the creation of streamlines on the whole brain involves the creation of duplicate paths and therefore redundancy in the final tractogram, while using the seeds only in some ROI involves an incomplete reconstruction and therefore not much used. In any case, the tractogram can be validated with respect to the use model, this involves the elimination of streamlines that do not connect two regions of the brain, as well as the elimination of streamlines that are too short. As described, the white matter has the function of connecting the regions of the gray matter. We can therefore say that the tractography algorithms create a large number of streamlines, which can connect or not different regions. In Figure 1.15 an example of tractogram used for the experiments reported in this thesis.

Tractography algorithms are divided into two main categories: *deterministic* and *probablistic*. Deterministic algorithms take only one possible direction for each voxel. So from the seed there will be only one streamline, that is a single possible path that follows the main orientation for each voxel, so the same trajectories are reconstructed from same seed points (Figure 1.16).

Instead, for probabilistic algorithms, the probability distribution on the principal diffusion direction is estimated at each voxel. The width in this distribution represents uncertainty in diffusion direction, which is due to factors such as the potential co-existence of many fibers directions within a voxel, image noise and subject motion in the scanner. The probabilistic tractography algorithm uses these local probability distributions to estimate the probability that a fiber pathway (or streamline) leaving the seed voxel will pass through any other voxel (Figure 1.17) [37].

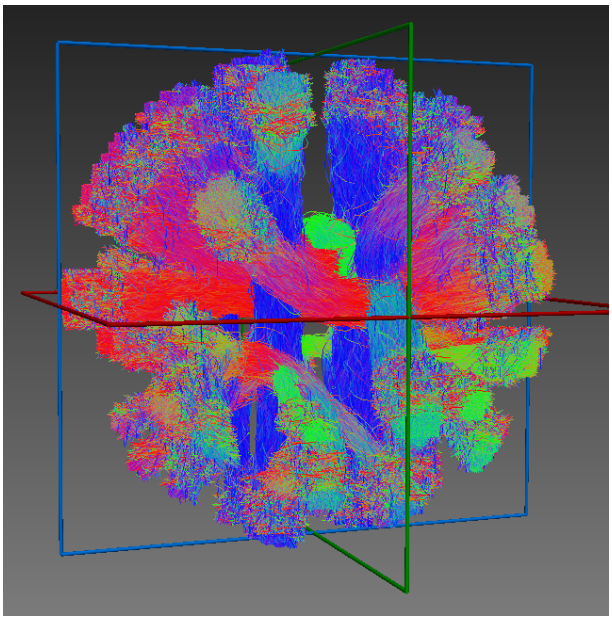


Figure 1.15: An example of tractogram used during the experiments. Image taken with MI-Brain tool [36].

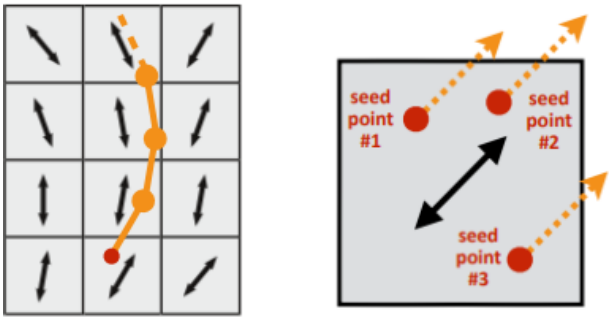


Figure 1.16: An example of deterministic tracking. Image taken from [14].

We remind that both the algorithms do not have the complete and perfect creation of the tractogram, the latter remains an approximation of the axons. Thanks to tractography, we can reconstruct the main bundles present in the white matter of the brain. These define the *connectome* of the human brain. *Connectome* is a graph, where *nodes* are the regions of the gray matter, while the *arcs* are the streamline bundles of the tractogram under examination.

The connectome matrix (Figure 1.18) is a square, symmetrical matrix that describes the number of how many streamlines connect two regions of

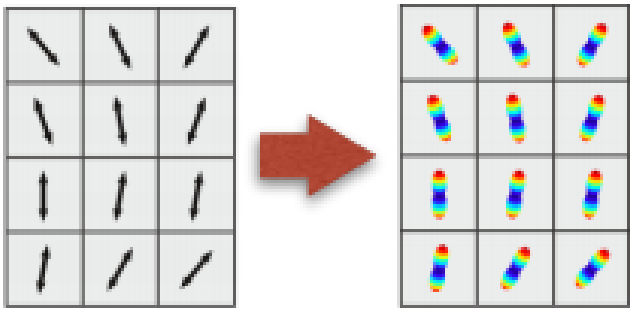


Figure 1.17: An example of probabilistic tracking. Image taken from [14].

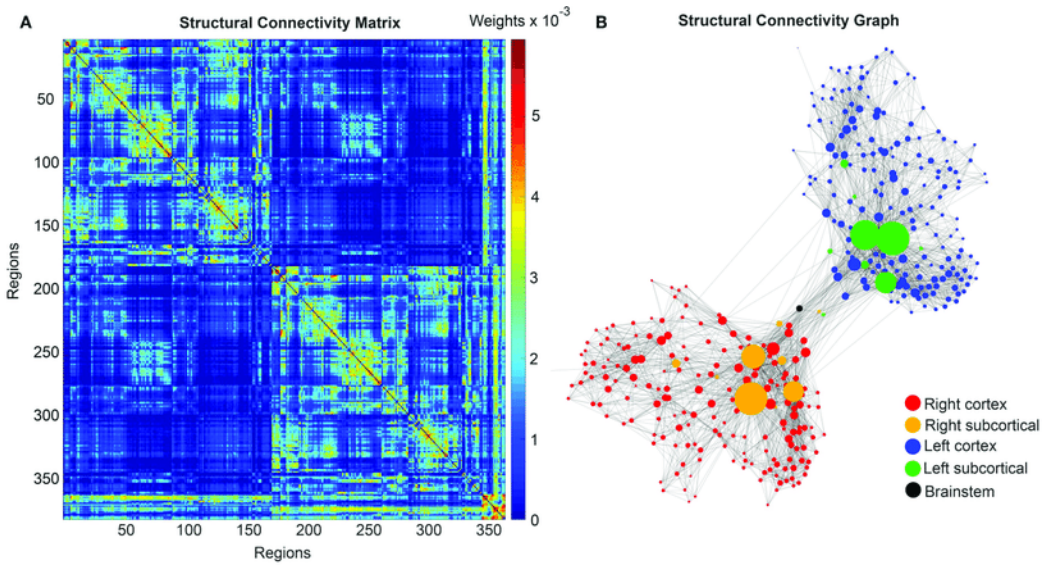


Figure 1.18: An example of connectome matrix. Image taken from [38].

the brain. Each cell of this matrix corresponds to the frequency of how many times those two regions are connected (*weights*). The main problem of the tractography is that the model is not perfect and it creates false positives (*specificity*). False positives are connections created by tractography which don't really exist in the human brain; they create errors in the interpretation of the connectome (Figure 1.19).

The goal is maximize the true positives and minimize the false positives connections in order to yield the classifier model better, the problem can

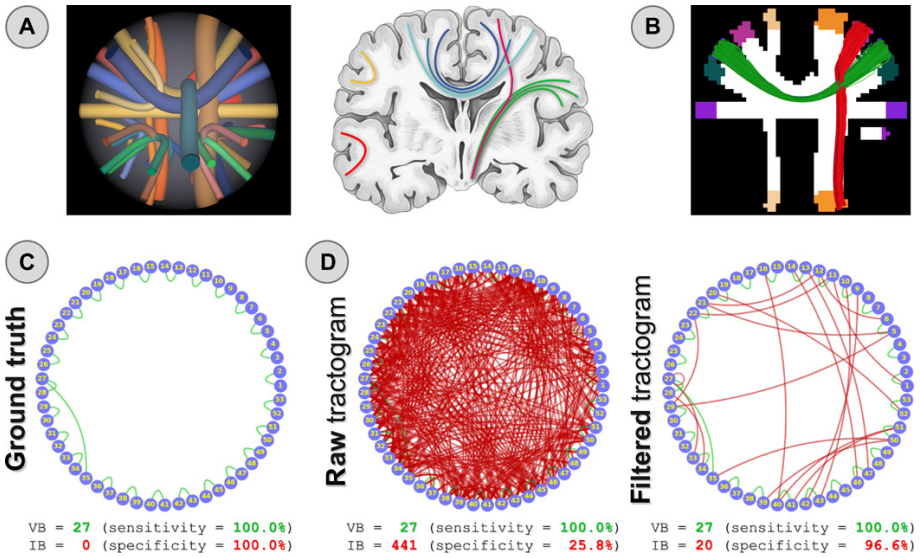


Figure 1.19: (A) Synthetic dataset (left), inspired by real anatomical bundles of the human brain (right), used to quantitatively evaluate our proposed method. White matter and gray matter masks used for tractography (B) and two examples of true-positive (green) and false-positive (red) bundles that can potentially be reconstructed with tractography. Ground truth connectivity represented as a graph (C): Blue circles correspond to the 53 gray matter regions shown in (B), whereas green and red arcs represent true-positive (VB) and false-positive bundles (IB), respectively. In (D), we compare the sensitivity and specificity of the estimated connectome before (left) and after filtering the tractogram with our COMMIT method (right). Image taken from [39]

.

be represented well with ROC (receiver operating characteristic) curve (Figure 1.20).

A ROC curve is a two-dimensional plot that illustrates how well a classifier system works as the discrimination cut-off value is changed over the range of the predictor variable [41]. The x-axis is the false positive rate for the predictive test, whereas y-axis is the true positive rate for the predictive test. Each point in ROC space is a true positive/false positive data pair of the predictive test. If the probability distributions for the true positive and false positive are both known, a ROC curve can be plotted from the cumulative distribution function [42].

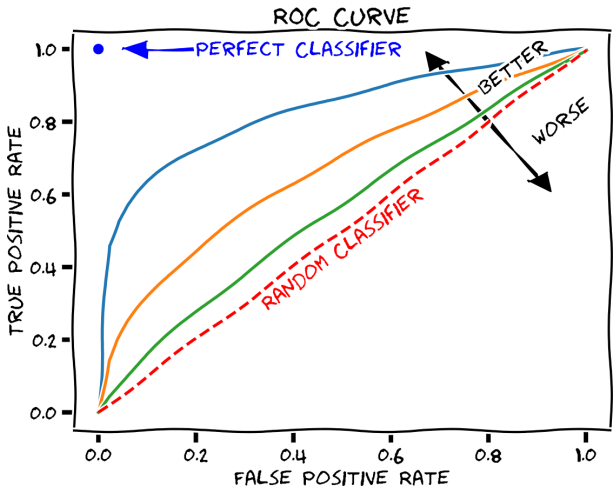


Figure 1.20: An example of ROC curve plot. Image taken from [40].

Using synthetic data as phantom, it is possible to easily identify which are the false positives or *invalid bundle*, as reference is made to a *ground truth*, which has 53 gray-matter regions, 27 white-matter bundles and zero invalid bundles (false positives) (Figure 1.19C).

It describes the known and certainly present bundles in the DWI model (Figure 1.21).

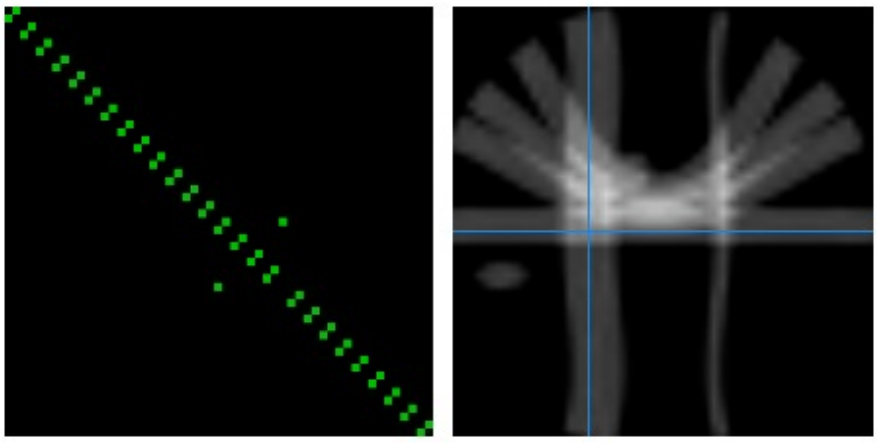


Figure 1.21: On the left we have the ground-truth connectivity matrix where there are only VB valid bundles; on the right, instead, we have the image Nifti file used for analysis.

Thus, the aim is minimize these invalid bundles connections in order to not introduce errors of interpretation and be as faithful as possible to the ground truth. For writing convenience, during this thesis, you might find abbreviated VB for *valid bundles* and IB for *invalid bundles*.

1.4 Microstructure Informed Tractography

Microstructure informed tractography is a relatively new area of research that aims at combining pieces of information using global optimization techniques in order to draw a quantitative map of the pipe network which, today, is not available [43]. In fact, several orders of magnitude separate the resolution achievable with dMRI from the actual size of the axons and each reconstructed trajectory has to be considered as representative of a coherent set of real anatomical fibers, the amount of which is not easy to assess [43]. As a consequence, nowadays the structural connectivity between brain regions is quantified by counting the number of recovered tracts or averaging a scalar map along them; either way, these quantities are only indirectly related to the actual underlying neuronal connectivity [44]. In order to remedy the false positives connections problem of the tractography and re-establish the link between tractography and information of the anatomical microstructure (Figure 1.22) is introduced COMMIT (*Convex Optimization Modeling for Microstructure Informed Tractography*) algorithm, developed by Professor Alessandro Daducci, University of Verona [45].

COMMIT uses models and global optimization techniques to combine the tractogram with information deriving from the microstructure. The great potential of computational diffusion MRI relies on indirect inference of tissue microstructure and brain connections, since modelling and tractography frameworks map diffusion measurements to neuroanatomical features [46]. Starting from an input set of candidate fiber-tracts, which can be estimated using standard fiber-tracking techniques, for example deterministic and probabilistic tracking, COMMIT models the diffusion MRI signal in each voxel of the image as a linear combination of the restricted contributions generated in every location of the brain by these candidate tracts. Then, COMMIT

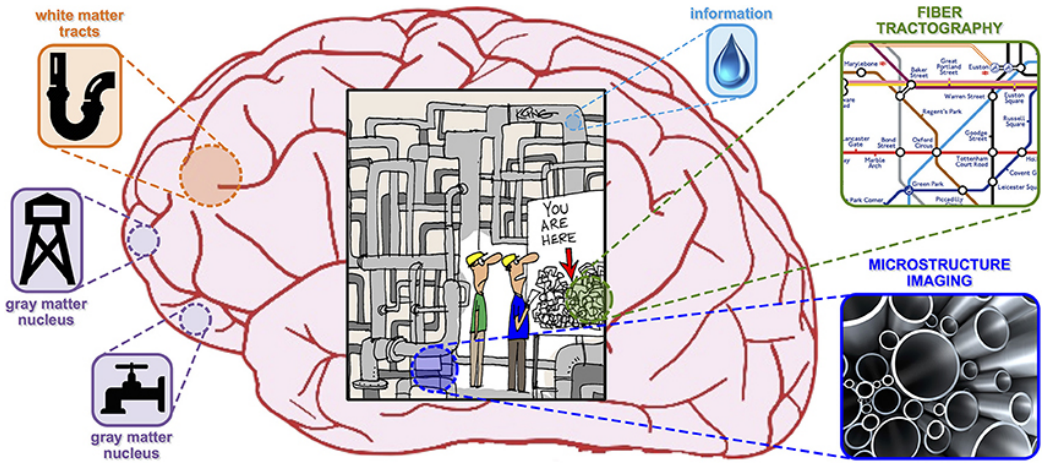


Figure 1.22: Microstructure informed tractography schema. Image taken from [43].

seeks for the effective contribution of each of them such that they globally fit the measured signal at best. These weights can be easily estimated by solving a convenient global convex optimization problem and using efficient algorithms. Results clearly proved the benefits of the proposed formulation, opening new perspectives for a more quantitative and biologically-plausible assessment of the structural connectivity in the brain [47]. COMMIT code is available in https://github.com/daducci/COMMIT. Recent studies [48] have proved the feasibility and the potential benefits of combining tractography with local microstructural features for mapping the connectivity. The common denominator for these methods is the estimation of a large set of candidate fiber-tracts using classical fiber-tracking algorithms, e.g., streamline, followed by a selection of a subset that best fits the acquired dMRI signal (Figure 1.23).

To solve these problems COMMIT tries to improve:

- *the quality of the reconstructed tractograms by combining them with the micro-structural properties of the tissue*
- *reduce the computational cost to meet the real needs of the applications*
- *guarantee to recover the global optimal solution*

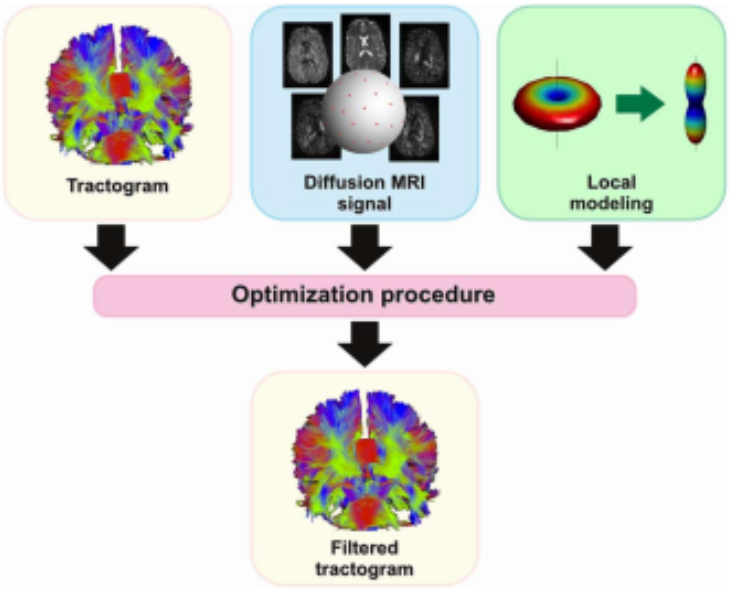


Figure 1.23: Block diagram of top-down strategies to combine tractography reconstructions with local properties of the tissue. Image taken from [45].

Approach to solve this problem consists of two steps. First, the candidate pathways are estimated from the data using classical fiber-tracking techniques, with the only requisite that these candidates represent a valid superset of the anatomically plausible tracts, i.e., true positives, possibly including also many false positives. Then, it seeks for the weight of each of them, i.e., real contribution or volume, by solving a global convex optimization problem (Figure 1.24) that exploits multicompartment models to explain the measured dMRI signal at best. Once the pathways have been estimated with any given tractography algorithm, the dMRI signal contribution of every tract must be mapped to each voxel of the image.

COMMIT needs as input the tractogram, estimated with a given tractography algorithm, and information of the anatomical microstructure. To do this, a multicompartmental model is used to characterize *neuronal tissue*, taking into account of both *confined water pools (intra axonal) that hindered ones (extra axonal)*, as well as the *partial volume with isotropic diffusion*. All this is described through a system of linear equations, as in the general form

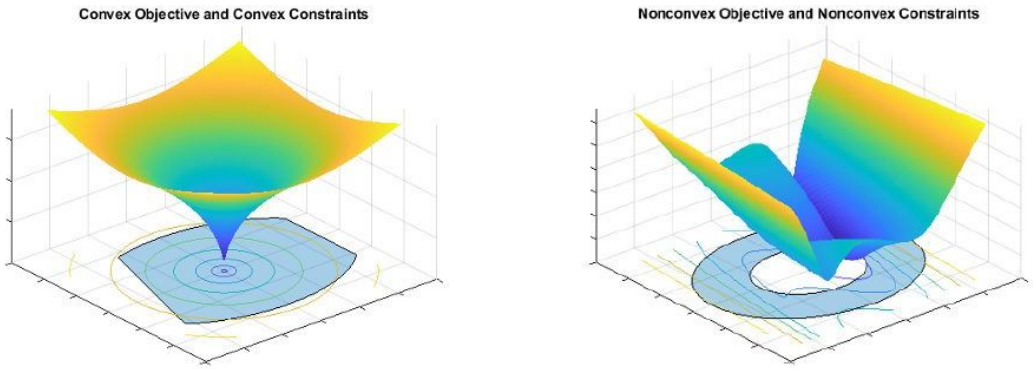


Figure 1.24: Convex vs non-convex cost function. Image taken from [49].

$\mathbf{A}x = b$, where the contribution with respect to a voxel is given by all the traits that cross the voxel itself [45]. The observation COMMIT model can be written in matrix form as:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \eta \,, \tag{1.6}$$

where $\mathbf{y} \in \mathbb{R}_{+}^{n_d n_v}$ is the vector containing the n_d q-space acquired in all n_v voxel (Figure 1.25), η defines both image acquisition noise and modeling error, $\mathbf{A} \in \mathbb{R}_{+}^{n_d n_v}$ is the observation matrix, i.e. the dictionary created by COMMIT, which explicitly models the multi-compartment model of the mapping ($\mathbf{A}$) in each voxel, while the positive weight values $\mathbf{x} \in \mathbb{R}_{+}^{n_c}$ are the contributions of the base functions in $\mathbf{A}$. The mapping matrix $\mathbf{A}$ is certainly very large, however many of its cells have a value of zero when a fiber crosses only a small part of a voxel of the acquired image. An example of this $\mathbf{A}$ operator is shown in Figure 1.26.

This operator describes a mapping function $\mathbf{A} : \mathcal{F} \rightarrow \boldsymbol{I}$ and is itself a block matrix, with the compartments *IC* (intra-axonal), *EC* (extra-axonal) and *ISO* (isotropic diffusion) [45]. The main purpose of COMMIT is to define the weight vector $\mathbf{x}$, to obtain the *fitting of the model*, in technical terms. Weights can be estimated by solving the following *non-negative least squares* problem (NNLS):

$$\arg\min_{\mathbf{x} \geq 0} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \tag{1.7}$$

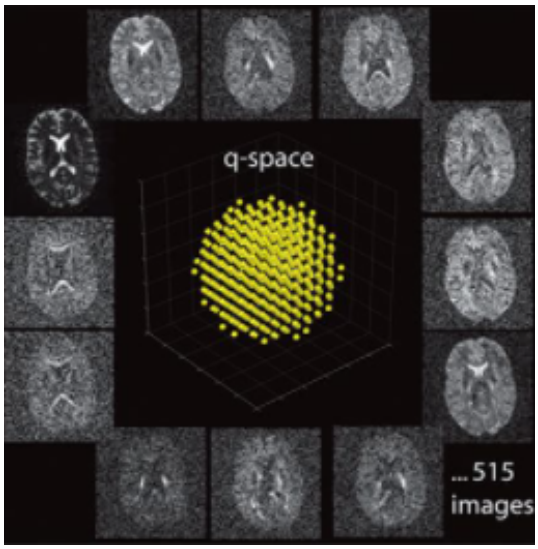


Figure 1.25: Representation of q-space model. Image taken from [50].

where $\|\cdot\|_2$ is the norm L_2 in $\mathbb{R}^n$.

As mentioned, the $\mathbf{A}$ matrix can be very large and therefore solving the system of equations may not be suitable. We therefore need a faster method to solve the problem of the equation (1.6), which is the basis of COMMIT. In many practical cases the system (1.6) can be underestimated, i.e. having a system with a number of unknowns greater than the number of measurements. This context can occur if the number of space measurements q in the input image is sub-sampling or if the set of candidate fibers is very large. In these circumstances, the inverse problem described by the equation (1.7) has not a single solution. However, if the coefficients to be recovered are known to be scattered, the use of L_1 norm is a very effective choice to find a solution to these ill-posed problems [45] [51]. So a solution for the system of equations model (1.6) can be solved by following the problem of L_1-minimization:

$$\arg\min_{\mathbf{x}\geq 0} \|\mathbf{x}\|_1 \text{subject to} \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2 \leq \epsilon \tag{1.8}$$

where $\|\cdot\|_1$ is the L_1 norm in $\mathbb{R}^n$ which is used to give sparsity in the solution $\mathbf{x}$ and the parameter ϵ is the parameter to manage the level of noise and error in the modeling phase. This parameter is an upper limit, therefore the

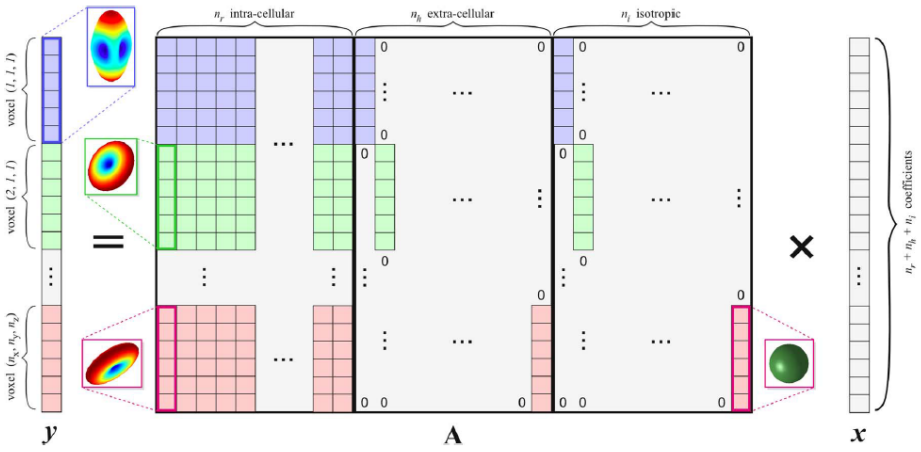


Figure 1.26: Representation of COMMIT model. Image taken from [45].

value of this norm L_1 must be lower, in order not to exceed the evaluation and weight of the streamlines. This formulation is known as *basic pursuit de-noise (BPDN)* [45].

COMMIT obtaining the vector $\mathbf{x}$ estimates in an appropriate way, following the anatomical information and the micro-structure of the brain, the tractogram, thus not stopping on the simple frequency of connection of regions by the streamlines. By doing so, the connectome itself becomes weighted, low values of connected regions can therefore be discarded, because they are defined as false positives. By doing so, we have a studied and coherent filter with respect to anatomy for the study of the human connectome.

Chapter 2

The problem of redundancy in tractography

The filtering methods described in the previous chapter require a large number of streamlines as input; however, as these algorithms are typically formulated as linear systems (see Eq. 1.6) this may introduce redundancy in the tractograms and, in turn, negatively impact on their performance [52]. The *aim of this thesis* was to characterize this problem in the case of COMMIT and study its impact on the quality of the connectivity estimates. Preliminary results of my work have been *published at the International Society for Magnetic Resonance in Medicine (ISMRM) conference* [53].

2.1 Redundancy and collinearity

In the COMMIT formulation, the signal contributions of a streamline are stored as a column in the matrix $\mathbf{A}$ (Figure 2.1,top); thus, if two (or more) streamlines follow similar trajectories, they are redundant for explaining the acquired dMRI signal and, more importantly, the corresponding columns tend to become linearly dependent (bottom panel). This phenomenon is called *collinearity* [52] and makes it difficult to interpret the estimated coefficients $\mathbf{x}$, i.e. the contributions of the streamlines.

The standard way to test if a linear system is affected by collinearity is

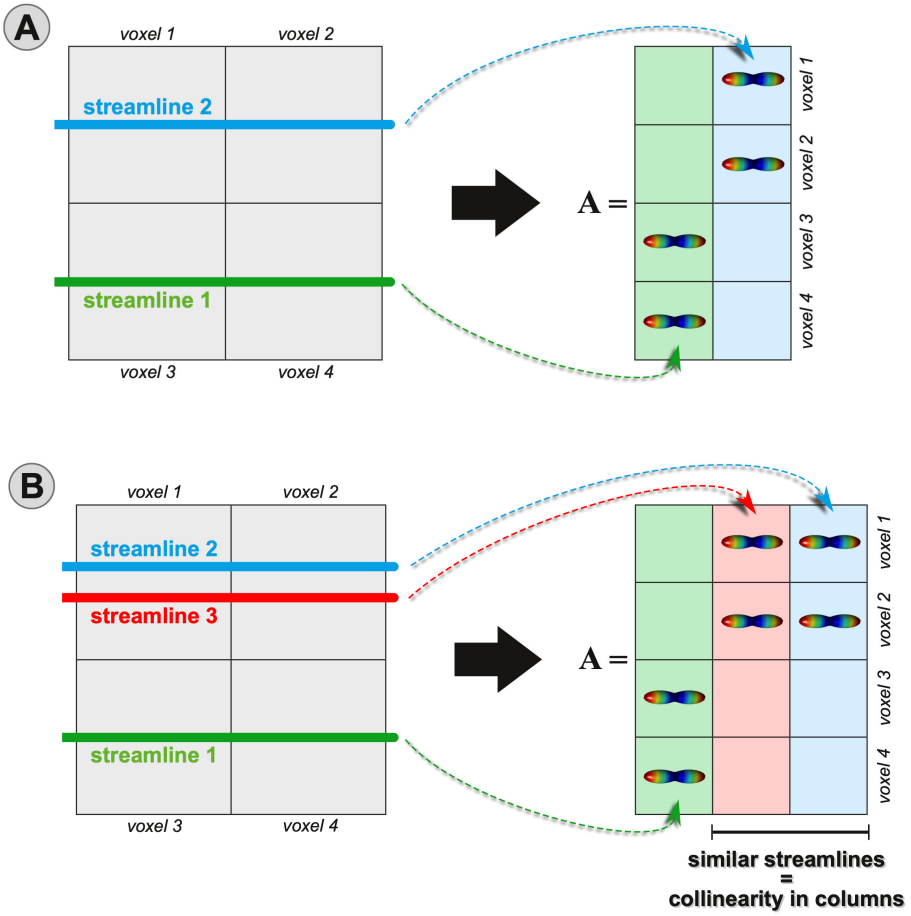


Figure 2.1: If streamlines follow similar trajectories, the corresponding columns in the matrix **A** tend to become linearly dependent, posing serious problems for the interpretation of the estimated coefficients. Image taken from [53].

to compute its condition number. The *condition number* is a property of a matrix, i.e. **A** in case of COMMIT, and it measures how much the output values can change, i.e. the contributions of the streamlines **x** in case of COMMIT, for small changes in the input data, i.e. the dMRI signal b measured in all voxels in the case of COMMIT. The condition number is useful to have indications about the sensitivity of the solution to perturbations in the data. In other words, when the condition number is large, the correct solution to the problem might be hard to find. A problem is said to be *well-conditioned* if its condition number is low, i.e. ≈ 1.0, whereas a problem with a high

condition number, i.e. $\gg 1.0$, is said to be *ill-conditioned* [51].

2.2 Condition number

The **condition number is defined** as:

$$k_p(A) = \|A\|_p \|A^{-1}\|_p \tag{2.1}$$

where $\|A\|_p$ is the p-norm of matrix A and $\|A^{-1}\|_p$ is the p-norm of the inverse of matrix A. It is worth noting that the *definition of the condition number depends on the choice of norm.*

A *generic p-norm* is defined as:

$$\|A\|_p = (\sum_{i=1}^{n} |A_i|^p)^{\frac{1}{p}} \tag{2.2}$$

In particular, the L_1 norm is simply the maximum absolute column sum of a matrix:

$$\|A\|_1 = \sum_{i=1}^{n} |A_i| \tag{2.3}$$

whereas L_2 norm is calculated as the square root of the sum of the squared vector values:

$$\|A\|_2 = \sqrt{\sum_{i=1}^{n} |A_i|^2} \tag{2.4}$$

Usually, the L_2 norm is used. In this special case, the formula for condition number becomes:

$$k_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \tag{2.5}$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are maximal and minimal singular values of $\mathbf{A}$ respectively, here we are exploiting the singular value decomposition (SVD) [54], in this way we can decompose the $\mathbf{A}$ matrix such as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{2.6}$$

where **U** is a mxm **unitary** matrix ($\mathbf{U}\mathbf{U}^\top = \mathbf{I}$), $\mathbf{\Sigma}$ is a mxn rectangular diagonal matrix with non-negative real numbers on the diagonal and **V** is a nxn unitary matrix (Figure 2.2). $\mathbf{\Sigma}$ matrix has singular values always greater or equal to zero and in ascending order on the diagonal. The condition number of **A** is defined as the norm of **A** times the norm of the inverse of **A**; the norm can be the usual L_2-norm (root-of-sum-of-squares) [2.4] or one of a number of other matrix norms [55].

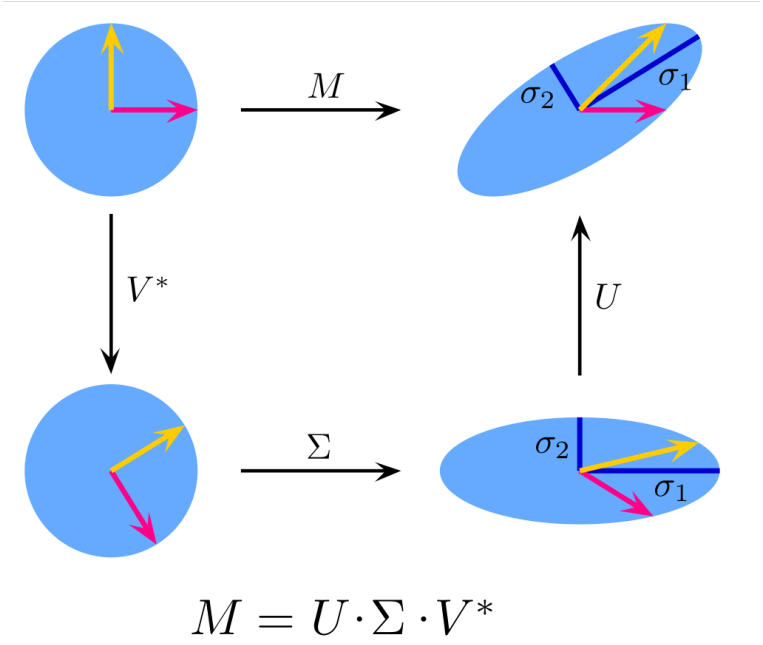


Figure 2.2: Geometric interpretation of SVD transformation and change of basis. Image taken from [56].

2.3 Removing redundancy using clustering

To mitigate the collinearity problem induced by the presence of redundancy among streamlines it is possible to use a combination of clustering of the streamlines (a detailed review of methods is given is chapter 3) and spatial blurring of their signal contributions. This new concept is called *"blurred streamlines"* and was recently introduced by our group in [53].

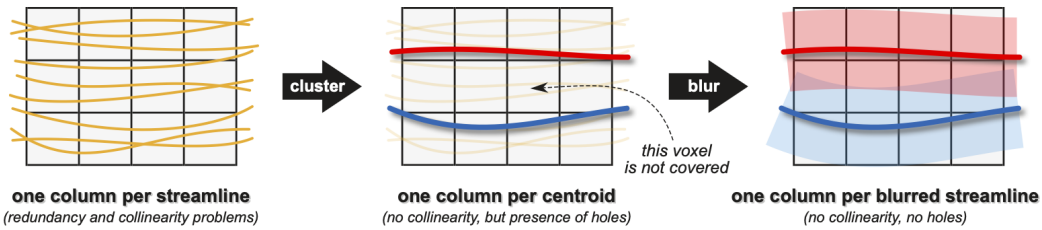


Figure 2.3: The new concept of blurred streamlines. Image taken from [53].

The idea behind the blurred streamlines is illustrated in Figure 2.3. First, the input tractogram (left panel) is clustered to minimize redundancy between its streamlines and prevent collinearity problems. The resulting centroids (red and blue in the middle panel) are streamlines representative of the estimated clusters and, by construction, are sufficiently dissimilar and likely generate non-collinear columns in the matrix **A**. However, the simplified tractogram consisting only of these centroid streamlines may not cover properly all voxels as the original tractogram and may not represent an optimal support set; in fact, if we observe the middle panel, we can notice some "holes" corresponding to some voxels that are not traversed by any streamlines and, as a consequence, in such voxels the signal cannot be properly represented (please recall that in COMMIT a streamline contributes to the signal only in those voxels it passes through). To mitigate this problem, we introduced the possibility to blur the signal contributions of each centroid (transparent areas in the right panel) and guarantee proper coverage of the white matter.

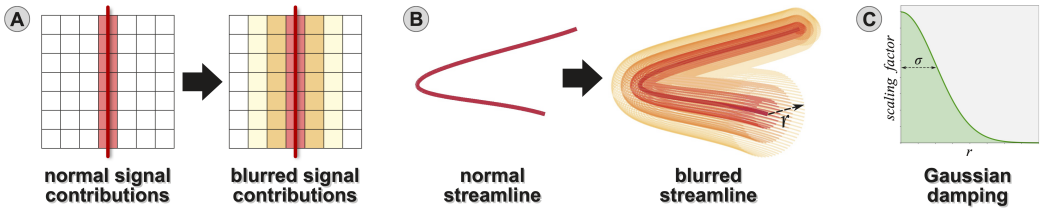


Figure 2.4: A *streamline* contributes to the signal only in those voxels it passes through, whereas a *blurred streamline* can contribute also to adjacent voxels. The signal contributions decrease according to a Gaussian distribution with standard deviation σ centered on the streamline itself, as function of the distance r. Image taken from [53].

With the term *"blurred signal contributions"* we mean that a streamline can contribute also to adjacent voxels (Figure 2.4A). An real example of application is shown in Figure 2.5. These additional contributions are scaled versions of the signal corresponding to the centroid streamline, as function of their distance r from this latter, according to a Gaussian distribution with zero mean and given standard deviation σ:

$$w = e^{\frac{-r^2}{2\sigma^2}} \tag{2.7}$$

. The sampling is done up to distance r_{max} where the Gaussian reaches the value 0.05:

$$r_{max} = \sqrt{-2\sigma^2 * \ln(0.05)} \tag{2.8}$$

As illustrated in Figure 2.4B, for each centroid streamline C we create a set of replicas by sweeping a series of N circles with radii $r \in r_1, r_2, ..., r_N$ along its path. All circles lie on a plane that is always orthogonal to C throughout the sweep and are subdivided in M sectors separated by an angle $\alpha = 2\pi/M$. Thus, each circle defines exactly M copies of C, one per sector, which are obtained by displacing radially all points of C from their initial position to a distance r and rotating them by an angle $k\alpha$ with $k \in 1, 2..., M$, about the center. This construction of the set of replicas of a streamline is based on the Frenet-Serret frames [57, 58] (Figure 2.6). In differential geometry, the Frenet–Serret formulas describe the kinematic properties of a particle moving along a continuous, differentiable curve in three-dimensional Euclidean space $\mathbb{R}^3$.

The signal contributions of a replica of C are computed as usual, i.e. considering only the traversed voxels, and the total signal contributions corresponding to the centroid C are obtained by summing those of all its replicas. However, as described before, the signal contributions of all replicas at radius r are scaled, as function of r, according to a Gaussian distribution centered on C itself with standard deviation σ (Figure 2.4C).

To summarize, the *blurred signal contributions of a blurred streamline corresponding to a centroid* C_i, which form the i-th column of the **A** matrix

are defined by the following formula:

$$A_i = \sum_{j=1}^{N} w(r_j) \sum_{k=1}^{M} S(C_i, i_j, \alpha_k) \tag{2.9}$$

where $\alpha_k = 2\pi k/M$ and $S(C_i, i_j, \alpha_k)$ are the contributions at radius r_j and section α_k, which are scaled by the factor $w(r_j) = exp(-0.5r_j^2/\sigma^2)$.

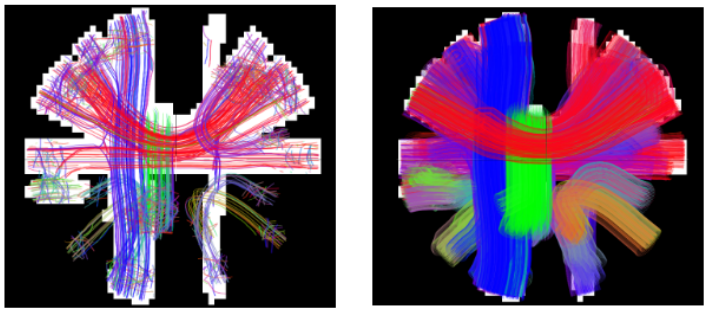


Figure 2.5: An example of blurring effect. On the left a deterministic clustered tractogram, on the right the same tractogram after applying blurring. Image taken from [59].

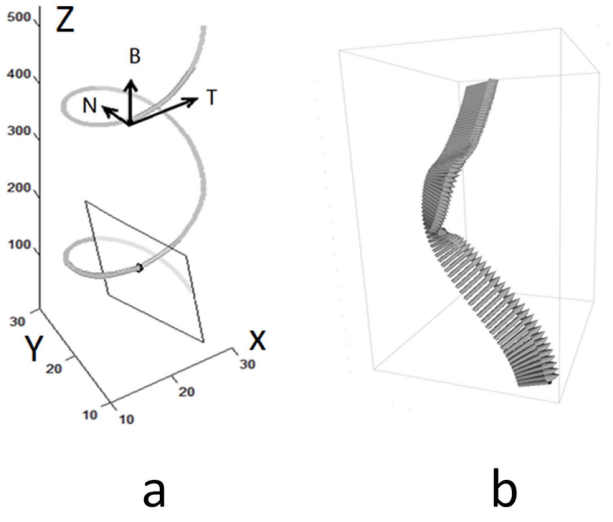


Figure 2.6: Illustration of the Frenet-Serret frames. Image taken from [60].

Chapter 3

Clustering techniques

Clustering can be considered as one of the most important unsupervised learning problems (Figure 3.1); it deals with finding a structure in a collection of unlabeled data. A cluster is a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters [61]. Clustering is a really important procedure as it determines the intrinsic grouping among unlabeled data [62].

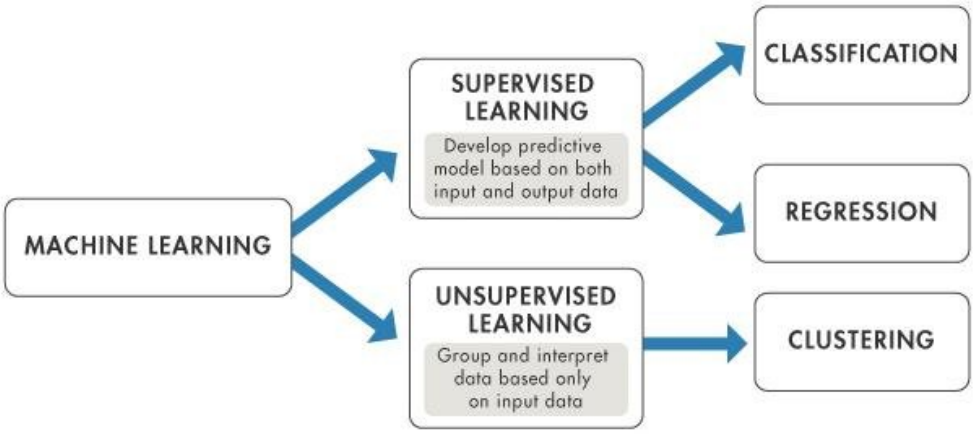


Figure 3.1: Differences between unsupervised and supervised learning. Image taken from [49].

3.1 Main clustering approaches

There are no criteria for a good clustering. It depends on the user and on which criteria they may use which satisfy their need. The various types of unsupervised learning clustering are:

- Centroid based (Partitioning methods)
- Connectivity based (Hierarchical clustering)
- Distribution based
- Density based (Model-based methods)
- Fuzzy

3.1.1 Centroid based

This approach is considered as one of the most simplest clustering algorithms, yet the most effective way of creating clusters and assigning data points to it [63]. The intuition behind centroid based clustering is that a cluster is characterized and represented by a central vector. It is consists of data points which are assigned to the respective clusters. These groups of clustering methods iteratively measure the distance between the clusters and the characteristic centroids using various distance metrics [64]. There are different types of distance, for example Euclidean distance, Manhattan distance or Minkowski distance (Figure 3.2).

We have to set intuitively or scientifically (using the Elbow method) (Figure 3.3) the number of clusters k, to begin the iteration of any clustering algorithm to start assigning the data points. One of the most important and widespread centroid-based clustering algorithm of this type is k-means.

The k-means algorithm is divided into three main stages:

- *Initialization*
- *Cluster assignment*
- *Update of the position of the centroid*

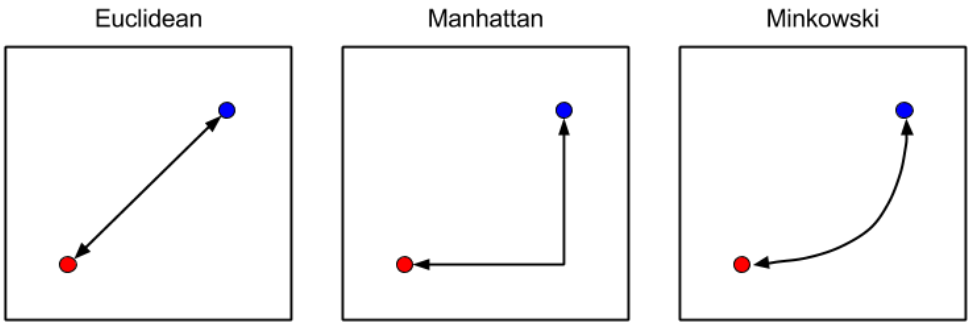


Figure 3.2: Some of the different metrics used to calculate the distance between two data points. Image taken from [65].

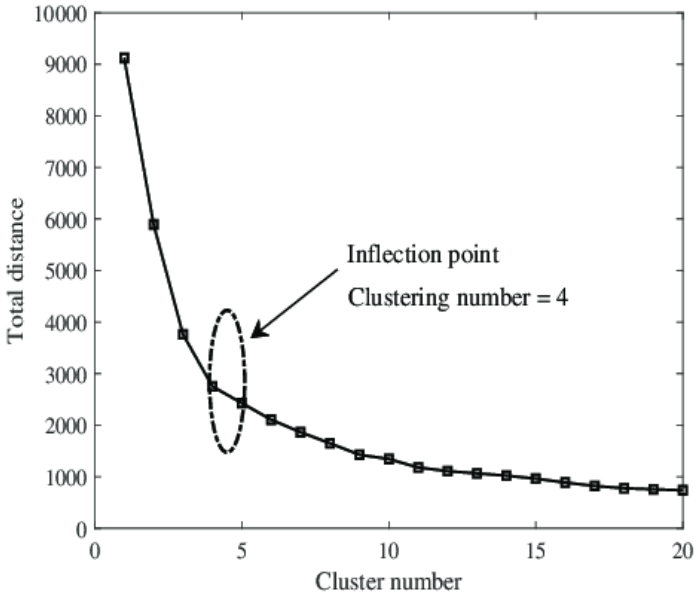


Figure 3.3: The Elbow method of k-means. Image taken from [66].

During the initialization we have to initialize the k-means by choosing the size of the data set and k initial centroids arranged randomly. In the second stage the algorithm analyzes each of the data points and assigns them to the nearest centroid, thus we calculate the distance, for example Euclidean (but we remind that there are other types of distance which we can be employed for computing the distance during this step [65]), between each data points

and each centroid. Each data points will then be assigned to the centroid whose distance is minimum:

$$\arg\min_{c_i \in C} dist(c_i, x)^2 \tag{3.1}$$

where there is a centroid in the set C (which includes all centroids), x are the datapoints and *dist* is the standard Euclidean distance, for instance. The last step consists in recalculating the average position of the centroids. The new value of a centroid will be the average of all data points that have been assigned to the new cluster. The algorithm is stopped when a point of convergence is reached, such that there are no more changes to the clusters [67].

Data: Data set D, Number of Cluster k, Dimension d:
C_i is the i-th cluster
(Stage 1, Initialization)
(C_1, C_2,...,C_k) = initial partition of D (Stage 2, Iterative phase).
Result: Input data cluster
repeat
  d_{ij} = distance between cluster i and cluster j;
  $n_i = argmin_{1 \leq j \leq k}\ d_{ij}$;
  Assign the element i to the cluster n_i;
  I recalculate the centroid of each cluster;
until *The centroids do not change*;
clusters output;

Algorithm 1: Pseudo-code k-means [68]

The complexity of the k-means algorithm is $\mathcal{O}(n*k*I*d)$, with n number of points, k the number of clusters defined initially, I the number of iterations to terminate the algorithm and d the size or cardinality of each element, then the number of features or dimensions for each point. Centroid-based algorithm is an efficient and simple clustering algorithm which, however, is sensitive to initial conditions and outliers (Figure 3.4).

Despite the flaws, centroid based clustering has proven its worth over hierarchical clustering when working with large datasets (Figure 3.5). These

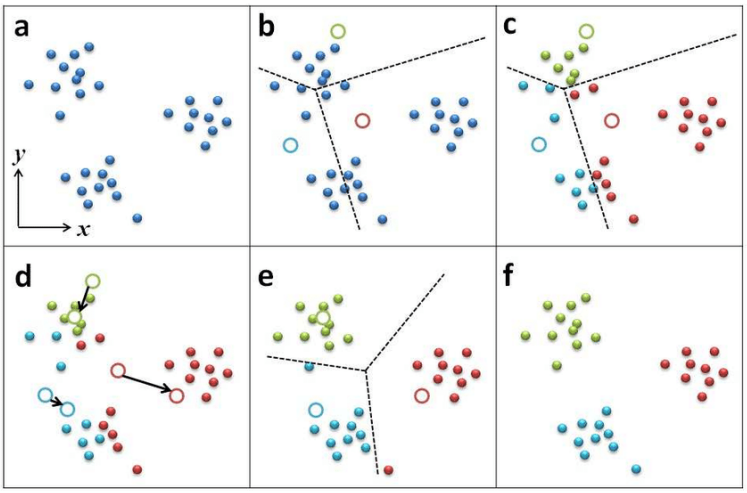


Figure 3.4: An example of the behavior of k-means algorithm. Image taken from [69].

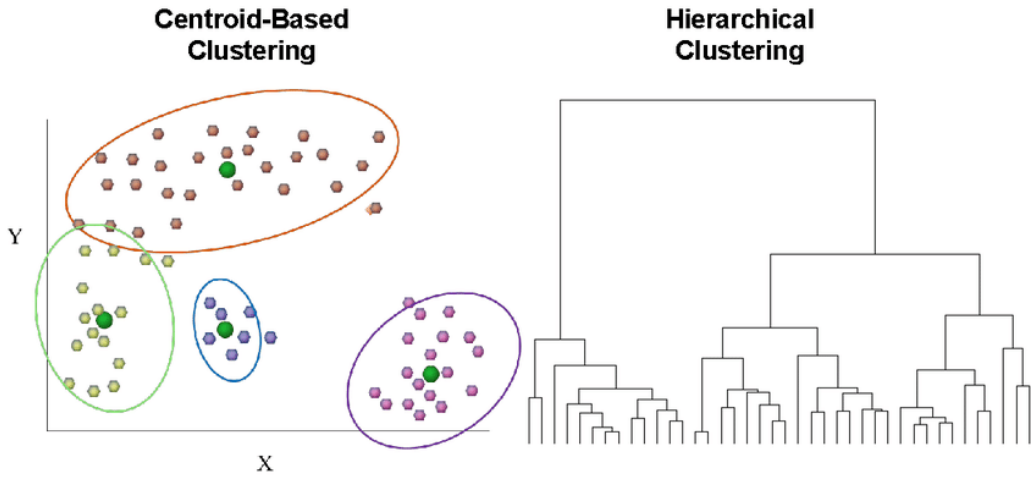


Figure 3.5: Differences between centroid and hierarchical approaches. Image taken from [70].

algorithms are widespread in many applications, as for example market segmentation, customer segmentation, text topic retrieval, image segmentation. All the tractography clustering algorithms seen in this thesis are centroid based.

3.1.2 Hierarchical

Hierarchical clustering, also known as connectivity-based clustering, is a method of unsupervised machine learning clustering which begins with a predefined top to bottom hierarchy of clusters [63]. It then proceeds to perform a decomposition of the data objects based on this hierarchy, hence obtaining the clusters. This method follows two approaches: top-down or bottom-up flow of creating clusters. The first is *divisive approach* and the second is *agglomerative approach*. The divisive approach of hierarchical clustering follows a top-down approach where we consider that all the data points belong to one large cluster and try to divide the data into smaller groups until a point beyond which there will be no further division of data points. Agglomerative is the contrary to divisive, where all the n data points are considered to be a single member of n clusters that the data is comprised into. We iteratively combine these numerous n clusters to fewer number of clusters, let's say k clusters and hence assign the data points to each of these clusters accordingly, this approach is a bottom-up one (Figure 3.6) [63].

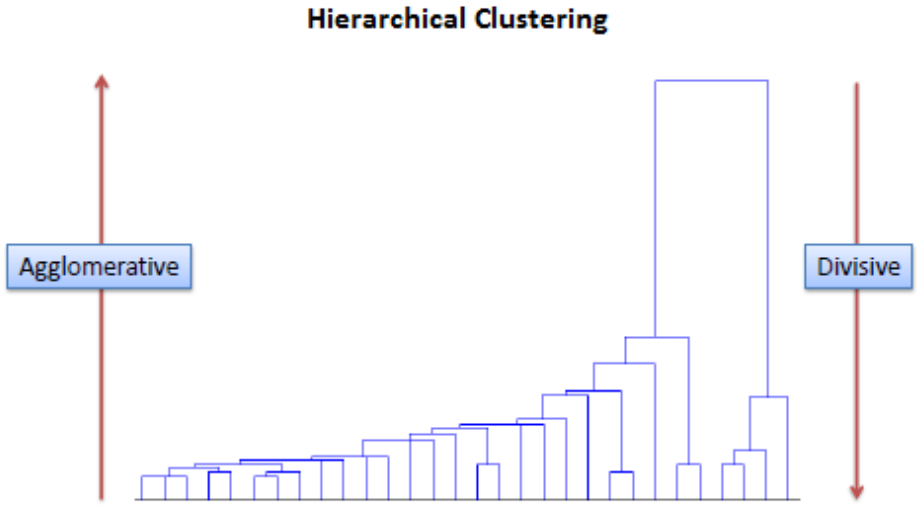


Figure 3.6: Two different hierarchical approaches. Image taken from [71].

3.1.3 Distribution based

Distribution-based clustering is based on distribution models. Clusters can be defined as objects belonging most likely to the same distribution, usually the Gaussian distribution. One prominent method is known as Gaussian mixture models (Figure 3.7). Here, the data set is usually modeled with a fixed number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to better fit the data set. This will converge to a local optimum.

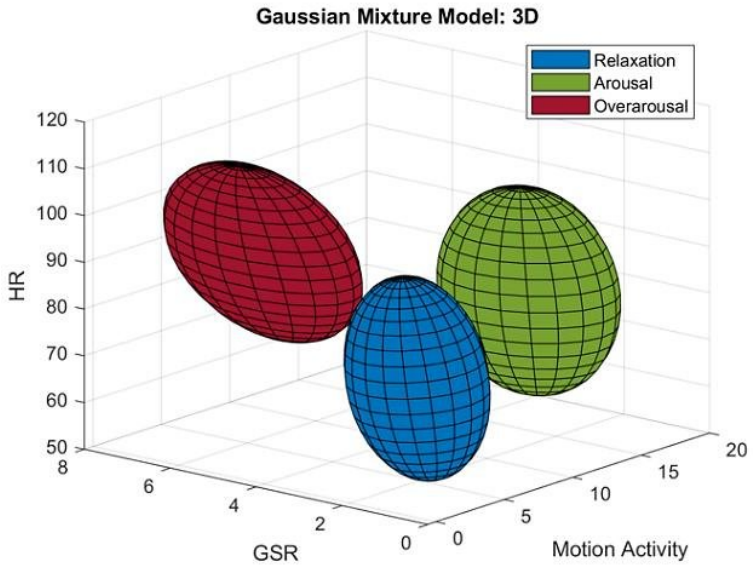


Figure 3.7: An example of Gaussian mixture model. Image taken from [72]

3.1.4 Density based

Density-based clustering methods take density into consideration instead of distances, based on the idea that a cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density. Clusters are considered as the densest region in a data space (Figure 3.8).

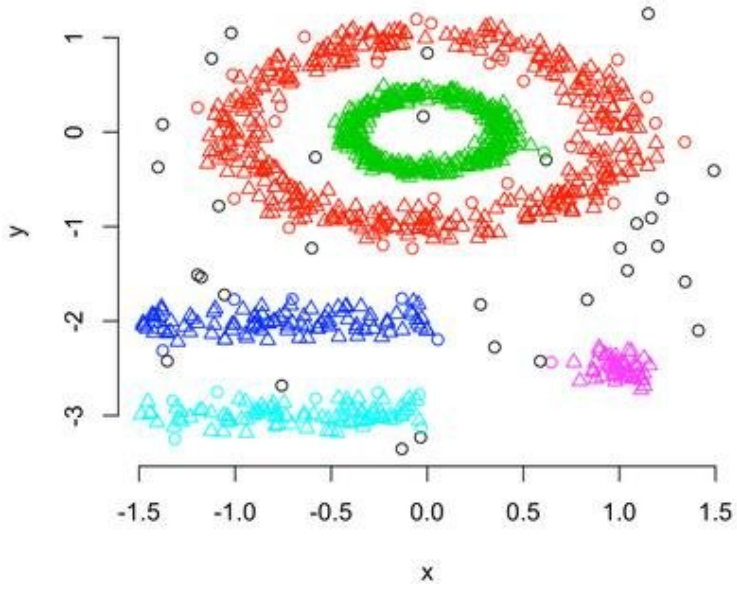


Figure 3.8: An example of density approach. Image taken from [73].

3.1.5 Fuzzy

In this form of clustering each data point can belong to more than one cluster [74]. The chance by which an element belongs to a given cluster is measured by membership coefficient that vary from 0 to 1. Fuzzy clustering can be used with datasets where the variables have a high level of overlap. It is a strongly preferred algorithm for Image Segmentation (Figure 3.9), especially in *bioinformatics*, where identifying overlapping gene codes makes it difficult for generic clustering algorithms to differentiate between the image's pixels and they fail to perform a proper clustering.

In unsupervised clustering algorithms given a data-set of n elements, each element must be compared with the remaining $n-1$ elements to measure the distance, in this way we have complexity $\mathcal{O}(n^2)$. This comparison also depends on the metric used and the parameters to make that comparison.

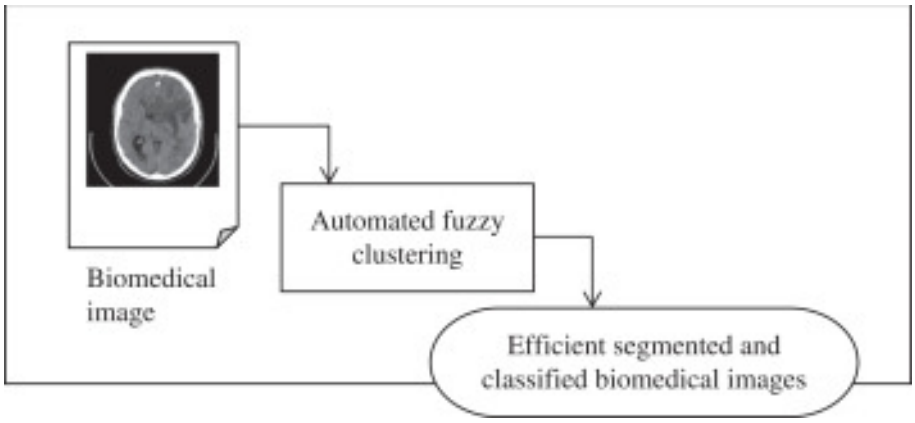


Figure 3.9: An example of application of Fuzzy approach [75].

3.2 Clustering tractography data

So far, we have seen the different categories of unsupervised learning clustering algorithms used in computer science and data analysis, but let's now analyze in detail the clustering algorithms that were specifically designed to work with tractography reconstructions, which were used and studied during this project. By definition, in unsupervised learning the focus is on creating a partitioning of the streamlines without knowing any labels. We start from the concept for which diffusion MRI data sets produce large numbers of streamlines which are hard to visualize, interact with, and interpret in a clinically acceptable time scale and tractographies have high levels of redundancy with many similar streamlines [76]. We compute cluster on input tractogram to *minimize the redundancy between its streamlines and prevent the onset of collinearity problems*. Collinearity occurs when independent variables in a regression model are correlated, i.e. the predictor variables in a logistic regression model are highly correlated [52]. If the degree of correlation between variables is high enough, it can cause problems when you fit the model and interpret the results. The resulting centroids are streamlines representative of the estimated clusters and, by construction, are sufficiently dissimilar and likely generate non-collinear columns in $\mathbf{A}$, one per centroid. The first algorithm which we applied on our input tractogram was QuickBundles [76].

3.2.1 QuickBundles

Quickbundles uses unsupervised learning to reduce in a simple and efficient way the number of streamlines, thus simplifying the process of bundles' segmentation and tractography exploration; indeed, it is faster than other main clustering algorithms. The code algorithm source is available here [1]. By the term bundle we mean here streamlines which are in close proximity according to a streamline-based distance, therefore, they have similar spatial and shape characteristics and not necessarily direct correspondence to neuroanatomical bundles (tracts) [76]. We remind that the unsupervised learning algorithms are of complexity $\mathcal{O}(n^2)$ where n the total number of streamlines: they require the calculation of all pairwise distances between streamlines in order to create a distance matrix but Quickbundles does not need to calculate all pairwise distances unlike most existing methods. Quickbundles uses a symmetric distance inter-streamlines function called which we call the minimum average direct-flip (MDF) distance:

$$d_{direct}(s,t) = d(s,t) = \frac{1}{K}\sum_{i=1}^{K}|s_i - t_i|, \tag{3.2}$$

$$d_{flipped}(s,t) = d(s,t^F) = d(s^F,t). \tag{3.3}$$

$$MDF = min(d_{direct}, d_{flipped}) \tag{3.4}$$

MDF is a symmetric distance function that can deal with the streamline bi-directionality problem; it works on streamlines which have the same number of points (Figure 3.10); the two streamlines are identified with s and t.

The direct distance $d_{direct(s,t)}$ between two streamlines s and t is the mean of the Euclidean distances between corresponding points. If a streamline of the data-set can be seen as a list of points in space, then we can describe a polyline as follows: $s = [s_1, s_2, s_3, ...s_k]$ in $\mathbb{R}^3$. We can define its corresponding streamline *flipped* or *inverse* as: $s^F = [s_k, s_{k-1}, s_{k-2}, ...s_1]$, where points are

[1] https://github.com/dipy/dipy/blob/master/dipy/segment/clustering.py

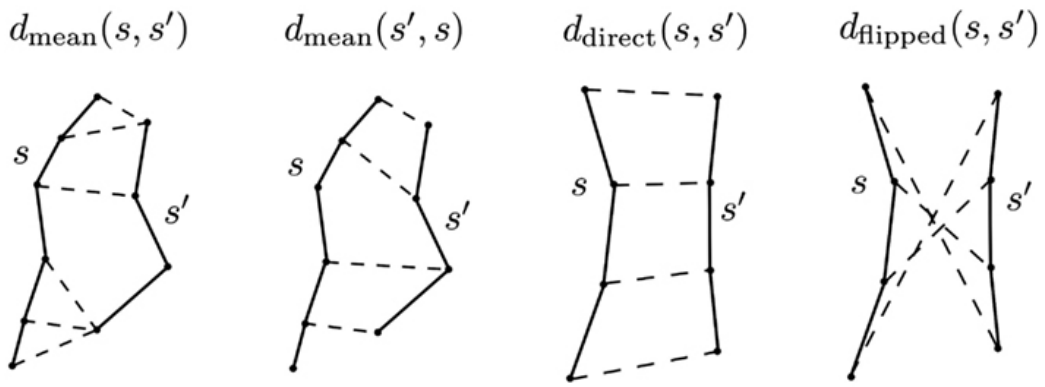


Figure 3.10: The components of both distances are shown; the streamlines are drawn with solid lines, and then with dashed lines we connect the pairs of points of the two streamlines whose distances contribute to the overall metric. Note that we cannot calculate the MDF between the streamlines on the left of the figure because they have different numbers of points. Image taken from [76].

considered in the reverse order of s, considering one or the other streamline flipped is indifferent for calculation purposes. The MDF distance is in fact a metric on the space of streamlines. As you can see, by using $|s_i - t_i|$ we refer to the Euclidean distance between two points, with regard to their dimensions. MDF will need to calculate $2K$ distances between points, for streamlines of length K. MDF distances are non-negative, are zero if and only if the two streamlines are identical, and symmetrical (all conditions of *triangle inequality* [77]). The main advantages of the MDF distance are that it is fast to compute, it takes account of streamline direction issues through consideration of both direct and flipped streamlines (Figure 3.11).

QuickBundles takes as input a data-set of streamlines with K control points each and outputs the clusters with the various polylines inside them and a new streamline considered as centroid. The algorithm saves everything in a data structure for each cluster, called *cluster node*. The streamlines are identified with indices $i = 1, ..., N$, where s_i is a $K \times 3$ matrix representing the i -th streamline. A *cluster node* is defined with a triple $c = (I, h, n)$, where I is the list of integer indexes $i = 1, .., N$ of the streamlines in that particular cluster , n is the number of streamlines in that cluster and h is the

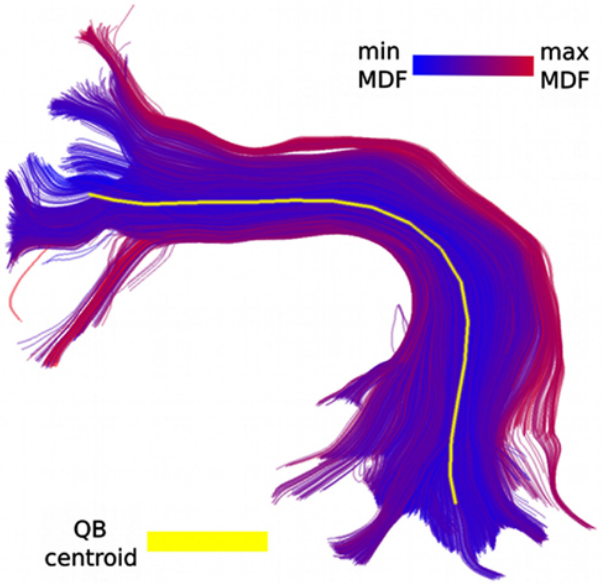


Figure 3.11: Color coding shows MDF distances from QB centroid to every other track in the bundle. Image taken from [76].

sum of the streamlines. Instead, h is a $K \times 3$ matrix that can be updated when a new streamline is added to the cluster:

$$h = \sum_{i=1}^{n} s_i \tag{3.5}$$

Where s_i is a $K \times 3$ matrix that represents the i -th streamline, $\sum$ represents the sum of matrices and n is the number of streamlines in the cluster. The cluster *centroid* can be calculated:

$$\nu = \frac{h}{n} \tag{3.6}$$

The centroid ν is a $K \times 3$ matrix. The *centroid* is a streamline created by an average of all the streamlines of the cluster and indeed is a representative of the cluster. This means that the generated streamline can be a streamline that does not connect regions in the brain, as it is different from all the others in the cluster. Here an example of application of this algorithm using DIPY [78] python library https://dipy.org/documentation/1.0.0./examples_built/

segment_quickbundles/.

One of the major benefits of applying QuickBundles is that it can provide meaningful simplifications and it can find features that were previously invisible or difficult to locate because of the high density of the tractography [76]. The QB representation is clearly shown in Figure 3.12 where every cluster is represented by a single centroid streamline.

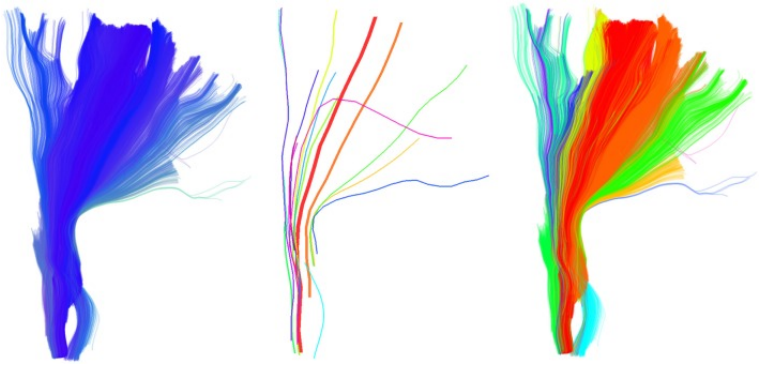


Figure 3.12: CST bundle, this bundle was labeled in the data sets provided by PBC (http://pbc.lrdc.pitt.edu).

We observe that only a few centroid streamlines travel the full distance from bottom to top and that they are many streamlines that are broken (i.e., shorter than what was initially expected) or highly divergent. Another interesting feature of QB is that it can be used to merge or split different structures by changing the clustering threshold (Figure 3.13). On the left we see simulated paths made from simple sinusoidal and helicoidal functions packed together. The color coding is used to distinguish the three different structures. With a lower threshold the three different structures remain separated but when we use a higher threshold the red and blue bundles are represented by only one cluster indicated by the purple centroid streamline.

The execution time of QB is affected by the following parameters: K, the fixed number of discretized points per streamline the clustering threshold th, which controls the heterogeneity of clusters and N the size of the subset of the tractography on which the clustering will be performed. When th is higher, fewer heterogeneous clusters are assembled, and conversely when is low, more clusters of greater homogeneity are created. The complexity of

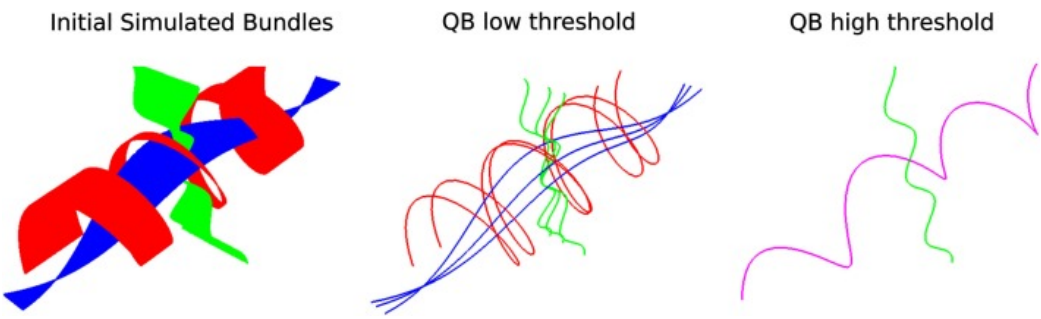


Figure 3.13: Left: 3 bundles of simulated trajectories; red, blue, and green consisting of 150 streamlines each. All 450 streamlines are clustered together using QB. Middle and Right: centroid streamlines using thresholds 1 and 8, respectively. At low threshold the underlying structure is reflected in a more detailed representation. At higher threshold, closer bundles merge together. Image taken from [76].

QB is in the best case linear time $\mathcal{O}(N)$ with the number of streamlines N and worst case $\mathcal{O}(N^2)$ when every cluster contains only one streamline. The average case is $\mathcal{O}(MN)$ where M is the number of clusters however because M is usually much smaller than N, we can neglect M and denote it only as $\mathcal{O}(N)$ as it is common in complexity theory. The following experiment to investigate this claim and we found empirically that the average case is actually $\mathcal{O}(N)$ for tractographies (Figure 3.14).

3.2.2 QuickBundlesX

QuickBundlesX shows a remarkable 20+X speedup over its predecessor who was until today the fastest clustering algorithm for streamlines [79]. The complexity of QuickBundles is $\mathcal{O}(kN)$ where k is the number of clusters and N is the number of streamlines. Because k is usually much smaller than N the complexity of the algorithm is near to $\mathcal{O}(N)$ with large distance thresholds (i.e. 30mm). However, when we start using small thresholds (i.e. 10mm), the number of clusters increases considerably and the complexity goes up accordingly. At same time k increases when N increases. Obviously, the more streamlines we add the more clusters will be created [79]. As in QB, also here, the distance metric used to compare the streamlines with

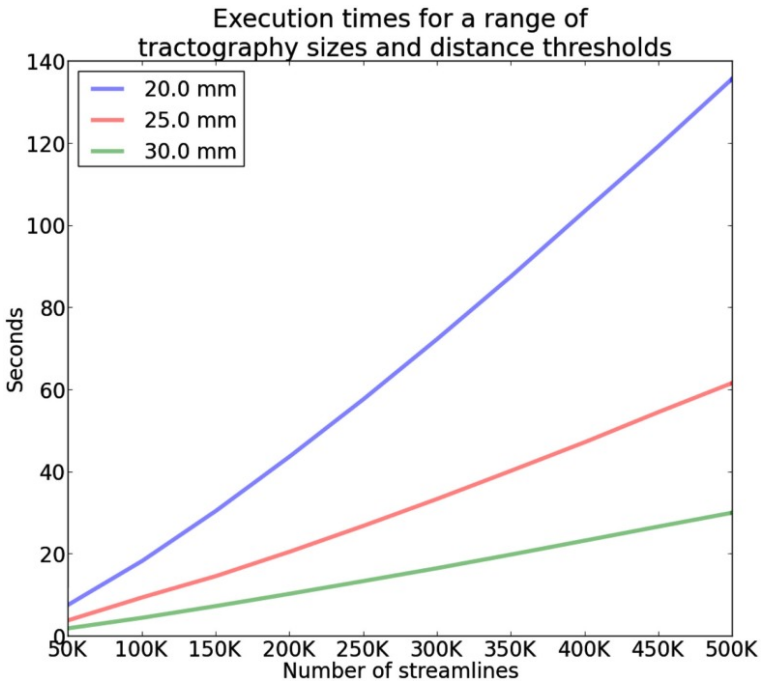


Figure 3.14: Time comparisons of QB using different clustering thresholds and different number of streamlines. Time increases linearly as the number of streamlines increases . Image taken from [76].

the centroids of the clusters is the minimum average direct and flip distance (MDF) (Figure 3.10). Each streamline is sequentially inserted in the tree from the root until it reaches the bottom.

The QBX algorithm works int this way, notice that points (1) and (2) are *mutually exclusive*:

1. a new child node is created
    - we move the next layer and the process is repeated until the last layer is reached
2. the streamline is inserted into an existing child node
    - we move the next layer and the process is repeated until the last layer is reached

A new child node is needed when the closest child node to the streamline

being inserted is too far (greater than the threshold of the current layer). Otherwise, we update the centroid of the closest child node and then repeat the process with the child node [79]. Using this tree you can always start from the root, find the centroid that is closest to you from the top layer and continue recursively searching the tree so now, with the QBX output tree, we have a better structure for searching in the space of streamlines [79].

3.2.3 FFClust

There is an another clustering algorithm that we studied, FFClust. We were attracted by the article of FFClust [80] in which its developers declared: "*FFClust is effective in the creation of compact clusters, with a low intra-cluster distance, while keeping a good quality Davies–Bouldin index* [81], *which is a metric that quantifies the quality of clustering approaches. Furthermore, it is about 8.6 times faster than the most efficient state-of-the-art method for one million fibers dataset. In addition, we show that FFClust is able to correctly identify atlas bundles connecting different brain regions, as an example of application and the utility of compact clusters*" [80]. This is an unsupervised clustering algorithm that groups the fibers into clusters, without recalculating the clusters, like such as k-means. FFClust implementation is publicly available from [2]. The algorithm uses a distance threshold to define whether a new fiber will be assigned to the closest cluster or will start a new cluster. It is based on the Minimum average Direct-Flip (MDF) (Figure 3.10) fiber distance. In input FFClust requires a tractography dataset of an individual subject consisting of a collection of fibers or streamlines, where each fiber is formed by 21 equidistant points in $\mathbb{R}^3$. Two orientations are possible: *direct* or *flipped*. Let Ts be a tractography dataset of an individual subject consisting of a collection of fibers or streamlines, where each fiber is formed by **21 equidistant points** in $\mathbb{R}^3$. FFclust uses Euclidean distance that we denote with $d_E(a_i, b_i)$ between corresponding points a_i and b_i of fibers a and b.

$$d_E(a_i, b_i) = \|a_i - b_i\| \tag{3.7}$$

[2]https://github.com/andvazva/FFClust

$$d_{EF} = d_E(a, b^F) = d_E(a^F, b) \tag{3.8}$$

$$d_{ME} = min(d_E(a, b), d_{EF}(a, b)) \tag{3.9}$$

The FFClust workflow is composed of four steps:

1. *Building point clusters*

2. *Generating preliminary streamline clusters*

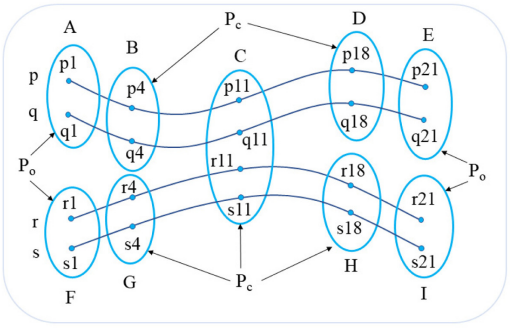
3. *Reassigning small preliminary streamline clusters*

4. *Merging candidate streamline clusters*

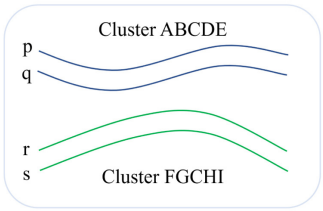
Building point clusters. This step aims at reducing the dimensionality of the input data by applying a partition clustering on a subset of streamline points. Now distance computations are performed on three dimensions fiber points instead of fibers formed by 21 points, where each point has three dimensions. The method uses a subset of five points (Figure 3.15a).

Minibatch K-means [82] (MK) was chosen as a partition algorithm because it is known to provide good quality, and low time and space complexity. Moreover, given that the clustering algorithm is applied on each streamline point independently, the number of clusters does not need to be the same in all streamline points. In fact, the proposed algorithm uses different numbers of clusters for streamline ending points and central points. We denote the number of clusters for ending points as K_{po}, and the number of clusters for intermediate and central points as K_{pc}. Using Elbow method [83] to find out the best number of clusters on each point [80].

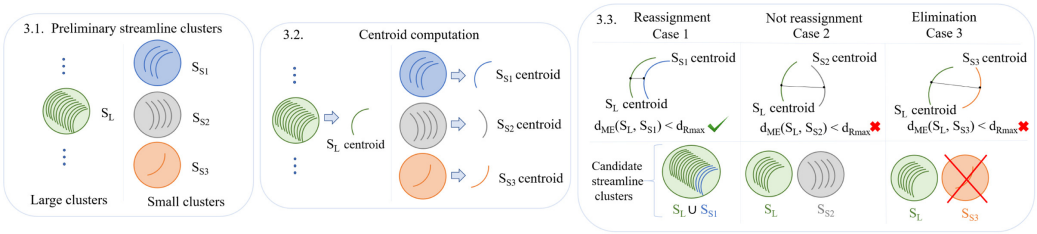
Generating preliminary streamline clusters. This step consists of building preliminary streamline clusters by grouping streamlines based on the membership labels of point clusters obtained in the previous step. A preliminary streamline cluster will contain all the fibers which points share the same point cluster labels.



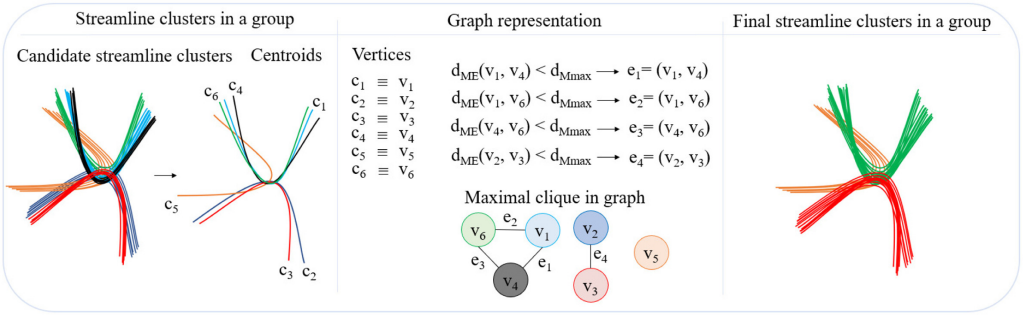
(a) STEP 1: Building point clusters.



(b) STEP 2: Generating preliminary streamline clusters.



(c) STEP 3: Reassigning small preliminary streamline clusters.



(d) STEP 4: Merging candidate streamline clusters.

Figure 3.15: FFCLust's workflow. (a) *Building point clusters.* MK is applied on the marked points. (b) *Generating preliminary streamline clusters.* (c) *Reassigning small preliminary streamline clusters.* (d) *Merging candidate streamline clusters.* Image taken from [80].

Reassigning small preliminary streamline clusters. This step reassigns small preliminary streamline clusters that could be separated from large clusters in the previous steps. A small cluster is reassigned to the nearest large cluster, given a maximum distance threshold and small clusters are discarded, so FFClust only discards small and isolated clusters, that are dissimilar to all the other large clusters. This is performed in its third step, which tries to reassign the small clusters to the largest clusters. All small clusters, with one or two streamlines that are not reassigned to a large cluster are eliminated. In order to do this, firstly divide the preliminary clusters in two sets based on their number of streamlines. A set S_L contains all preliminary clusters with number of streamlines equal or greater than 6, set S_S contains all preliminary clusters with 5 or fewer streamlines. Centroids for each preliminary cluster in both sets are computed as the arithmetic mean of each streamline point. Then, a preliminary cluster in set S_S is reassigned to the closest preliminary cluster in set S_L, only if the distance between their centroids is below the threshold d_{Rmax} if $d_{ME}(a, b) < d_{Rmax}$, otherwise, corresponding clusters are not reassigned.

Merging candidate streamline clusters. In this step we have to merge candidate clusters which still might be close based on a maximum distance parameter d_{Mmax}. The candidate cluster centroids in each group are merged based on the maximum distance parameter d_{Mmax}. If candidate cluster centroids are below d_{Mmax} for a maximumEuclidean direct or flipped distance, then such clusters are merged. The graph representation considers that each candidate cluster centroid is a vertex, u, in an undirected graph, $G(u, v)$ and there is an edge, e, between two vertices, u and v, only if the cluster centroids they represent are below a maximum Euclidean distance threshold d_{Mmax}, that is, only if $d_{ME}(u, v) < d_{Mmax}$ (3.9). For example in Figure 3.15d there are six candidate cluster centroids $(c_1, c_2, c_3, c_4, c_5, c_6)$ represented with corresponding vertices $(v_1, v_2, v_3, v_4, v_5, v_6)$. In this case, there are four edges (e_1, e_2, e_3, e_4), which exist only because the distance threshold is satisfied (3.9) [80]. Now let's take advantage of the concept of *clique*, where each group consists of vertices where all pair of vertices are connected by an edge. We have to the find maximal cliques, which are cliques that cannot grow by

adding another vertex, for example in Figure 3.15d we have 3 maximal cliques and the method sorts all maximal cliques by decreasing size and merges all candidate clusters represented in cliques having at least two vertices, which clusters have not been previously merged. During FFClust algorithm we can set three configurable parameter:

- *the number of clusters* (K_{pc} and K_{po}) for each of the five streamline points on which the MK algorithm is applied (1)
- *the maximum Euclidean distance threshold* (d_{Rmax}) for the reassignment of small to large preliminary clusters (3)
- *the maximum Euclidean distance threshold* (d_{Mmax}) for merging candidate clusters into final clusters (4)

Here a comparison results between FFClust algorithm and the other main clustering algorithms in tractography (Figure 3.16).

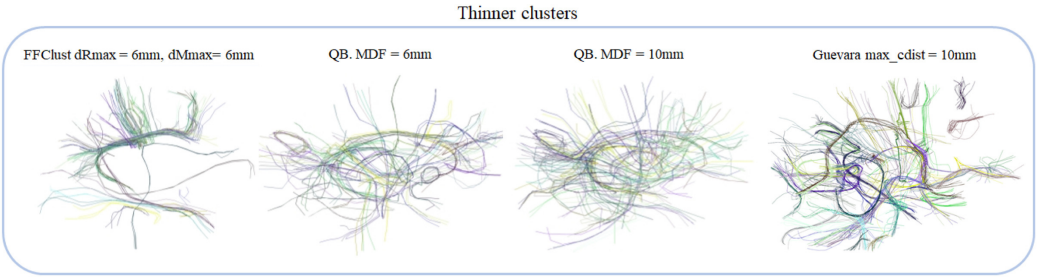


Figure 3.16: FFClust's results, QBmdf6, QBmdf10, and Guevara with the 50 thinner clusters. Visual inspection shows that the methods provide similar clusters, however QB and Guevara clusters seem to be more scattered than clusters obtained by FFClust. Image taken from [80].

Now let's analyze the complexity of this algorithm. Obviously in order to analyze the total complexity of the algorithm we have to calculate the partial complexity for each step of the algorithm. In the STEP 1 we have $\mathcal{O}(tKDN)$, where N is the number of elements, D is the dimensionality of the elements, K is the number of clusters and t is the number of iterations or until convergence. In the STEP 2 we have $\mathcal{O}(N)$, this is the fastest step of all and, if we desire, this step is also implemented in parallel computation

thanks to OpenMP framework [84]. The time complexity of this step is $\mathcal{O}(\|S_i\| * \|S_l\|)$ where $\|S_i\|$ and $\|S_l\| \gg N$ because arre centroids and not fibers. In the last step the complexity is determined by maximal clique, also this step is parallelized with OpenMP [84].

Chapter 4

Experiments and Results

4.1 Materials and methods

4.1.1 Data

For the experiments I have used the synthetic phantom of *International Symposium on Biomedical Imaging* (ISBI 2013) [39]; the dataset is shown in Figure 4.1.

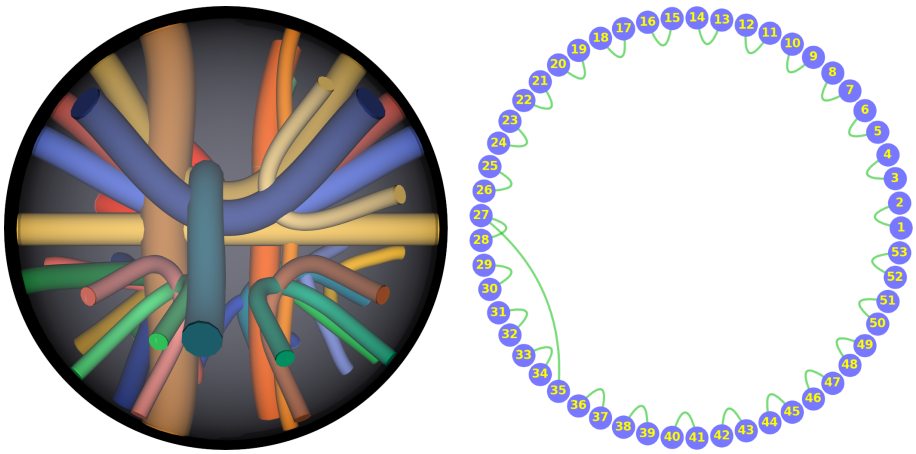


Figure 4.1: On the left, synthetic phantom in 3D view. On the right, ground-truth graph used during the experiments.

It is a mathematical model that describes bundles of streamlines which can simulate some connections actually present in the human brain. These

mathematical models are used to test algorithms and to verify their correctness and functionality. This phantom is made up of 27 bundles, true positives (VB), which connect a total of 53 regions (sec 1.3). Note that, the ground truth graph consists of only green arcs (true positives), there is no false positive, indeed, it has no red arcs (Figure 4.1,right). Furthermore, it is used a white matter (Figure 4.2), which describes the area considered in the model.

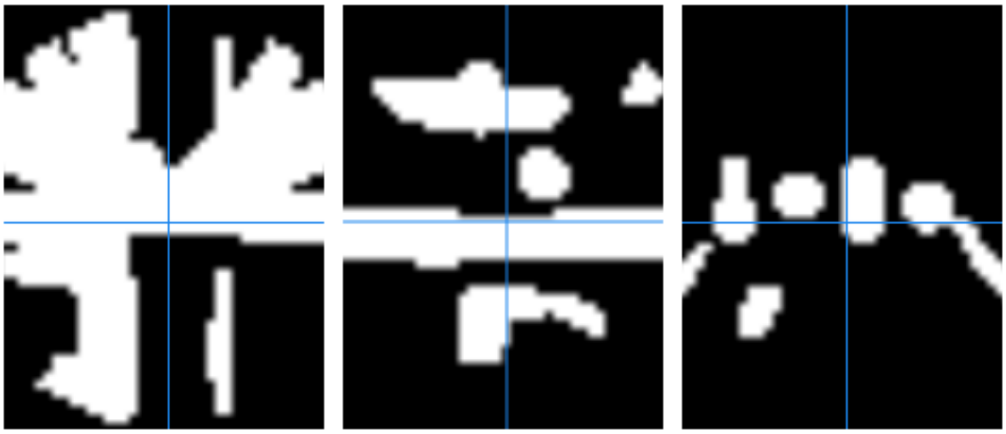


Figure 4.2: White matter model used for experiments. Left panel: Coronal view; Middle panel: Axial view; Right panel: Sagittal view. Image taken using Mango tool [85].

4.1.2 Streamlines reconstruction

Tractography consists in creating the tractogram, the tool used during this pipeline is MRtrix3 [86], so commands of this tool will be used to perform the tasks.

The first command used is *tckgen* [87], it performs streamlines tractography, by default, this command produces a fixed number of streamlines, by attempting to seed from new random locations until the target number of streamlines have been selected or the maximum number of seeds has been exceeded in the mask image (WM image). During the experiments, I always use two different types of algorithms for tractography: *deterministic* and *probabilistic*. In MRtrix3 the options are:

- *SD_STREAM*: Streamlines tractography based on Spherical Deconvolution (SD) [88]. A deterministic algorithm that takes as input a Fiber Orientation Distribution (FOD) image represented in the Spherical Harmonic (SH) basis. At each streamline step, the local (trilinear-interpolated) FOD is sampled, and from the current streamline tangent orientation, a Newton optimization on the sphere is performed in order to locate the orientation of the nearest FOD amplitude peak [86];

- *iFOD2*: Second-order Integration over Fiber Orientation Distributions. A probabilistic algorithm that takes as input a Fiber Orientation Distribution (FOD) image represented in the Spherical Harmonic (SH) basis. Candidate streamline paths (based on short curved "arcs") are drawn, and the underlying (trilinear-interpolated) FOD amplitudes along those arcs are sampled. A streamline is more probable to follow a path where the FOD amplitudes along that path are large; but it may also rarely traverse orientations where the FOD amplitudes are small, as long as the amplitude remains above the FOD amplitude threshold along the entire path [86].

The second step in tractography is *tck2connectome* command [89], it generates a connectome matrix from a streamlines file and a node parcellation image.

The last command, as regards the tractography, is *connectome2tck* command [90], it extracts streamlines from a tractogram based on their assignment to parcellated nodes. Moreover, there are some parameters during tractography step that must be set:

1. *number of streamlines selected*
2. *type of algorithm*
3. *tracking_step*
4. *tracking_angle*

where the *number of streamlines selected* defines the number of streamlines to generate; *type of algorithm* defines which tractography algorithm to use, for

example deterministic or probabilistic; *tracking_step* referring to the equation (1.5) indicates the size of the Δ, measured in mm; *tracking_angle* defines the maximum angle, expressed in degrees, that can exist between two sections that the algorithm decides to connect.

4.1.3 Clustering per bundle

Once the tractography step is finished, clustering is applied. It starts with a first division of the initial tractogram, created either with deterministic or probabilistic algorithm. Different *.tck* files will be created containing the various bundles with respect to the connected regions. These small data sets do not require a lot of time computation to be clustered, so QuickBundles is used [76]. Finally, the individual clustered bundles are grouped together to create the tractogram of the entire model, but with a lower number of streamlines and with all the bundles compared to the initial data-set (Figure 4.3).

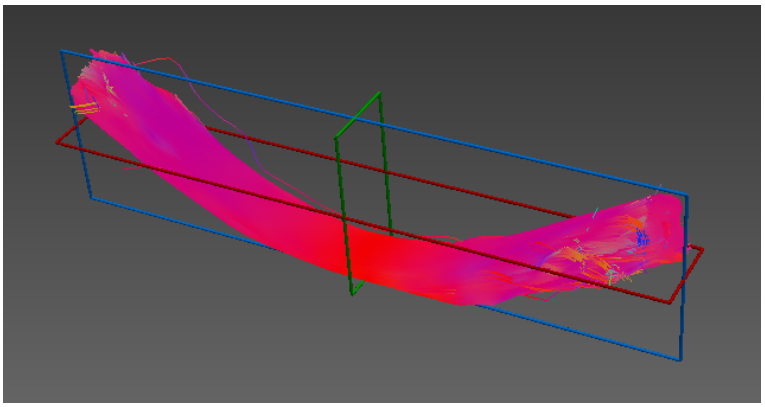


Figure 4.3: An example of reconstruction bundle 44-45 *tck file* from deterministic tracking algorithm used during the experiments. Image taken using MI-Brain tool [36].

4.1.4 Evaluation of the reconstructions

To evaluate the quality of the tractograms created during the tractography, I compared the reconstructed streamlines with the ground-truth using different metrics:

- VB
- IB
- RMSE

where RMSE is the *root-mean-square error* of the fit, a standard way to measure the error of a model in predicting quantitative data [91]; the formula is:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{4.1}$$

where $\hat{y}_i$ are the predicted values, y_i are the observed values and n the number of observations.

Furthermore, we used two metrics to evaluate also the computational costs and performances of the pipeline implemented:

- Time
- RAM

where Time is the computational time which is required to run the entire pipeline (in minutes) and RAM, instead, is the memory used during the execution of pipeline (in GigaBytes), keeping in mind that the entire pipeline consists of the following steps: *Tractography* $\rightarrow$ *Clustering* $\rightarrow$ *COMMIT* and *COMMIT + blur*. In table 4.1, you can find the metrics used in the accuracy experiments for the evaluation of the reconstructions analysis.

4.1.5 Libraries and software

All scripts and codes that were used for the analysis to get the results, which I will show shortly, have been implemented in Python3 [92] using different Python libraries to help different needs. Some of them were used for data management (i.e. Numpy [55], Scipy [93], Pandas [94]), others for plotting (i.e. Matplotlib [95], Seaborn [96]) and, lastly, for neuroimage processing (i.e. Dipy [78], Nibabel [97]). Moreover, during the analysis, I have used two different tools for the visualization of the files with which I worked. For

EXPERIMENTS		
	SD_STREAM	iFOD2
number of streamlines	1 million	1 million
tracking_step	0.25mm	0.50mm
tracking_angle	60°	45°
clustering thresholds	[0.0, 0.10, 0.20, 0.40, 0.50, 0.75, 1.0, 1.25, 1.5]	[0.0, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0]
sampling_step	0.25mm	0.25mm

Table 4.1: Table of accuracy experiments.

the neuroimages (Nifti files) I have used Mango [85] and for the streamlines visualization in *tck* format [86], I have used Mi-Brain [36] developed by Imeka.

4.2 Results

4.2.1 Collinearity and conditioning of the problem

As explained previously (sec 1.4), COMMIT is global convex optimization problem that exploits multi-compartment models to explain the measured dMRI signal function (f) at best. The *condition number* k_p gives an indication of the tightness of the bounds on the curvature of f and therefore of the difficulty to optimize it: the bigger the value of k_p, the slowest the convergence of the algorithm [98]. Thus, the smaller the condition number, the faster the function to reach a convergence value. The first analysis to be carried out is to understand whether COMMIT quickly converges to a value and, if the variant COMMIT blur, helps in terms of speed in reaching the convergence value and how much clustering affects the condition number.

Computation of the condition number of A

As first experiment for evaluating the effects of ill-conditioning due to streamlines redundancy, I computed the condition number of the linear operator $\mathbf{A}$

EXPERIMENTS	
	SD_STREAM
number of streamlines	1 million
tracking_step	0.50mm
tracking_angle	45°
clustering thresholds	[0.0, 0.1, 0.2, 0.4, 0.6]
blur sigma	[0.0, 0.5, 1.0, 2.0, 3.0]
sampling_step	0.25mm

Table 4.2: Table of condition number experiments.

of COMMIT (1.7) after different clustering levels. However, as the size of the matrix **A** can be prohibitive, in these experiments I extract only a bundle of streamlines in order to reduce the size of **A** to a manageable value. It is possible to adopt this solution because the streamlines redundancy, whether it exists, it is only between similar streamlines, hence, those that form the same bundle.

At this point, I compute the condition number of the **A** matrix. The command used for the condition number is *np.linalg.cond*(A), so I use the Python3 [92] library Numpy [55]. Note that, if it is not inserted any second parameter, for computing the condition number the function will use the L_2 norm, computed directly using the SVD (Figure 2.2). To visualize the results, it is used a particular plot called *heatmap*, which is a graphic representation of data whose values are represented by colors [99]. Thanks to the experiments carried out [Table 4.2], looking at Figure 4.4, note that there is an higher condition number for clustering threshold equal to 0.0 and 0.1mm for all blur sigmas and also to clustering threshold 0.2mm, but only for blur sigma equal to 0.0. Whereas, for clustering thresholds greater than 0.2mm (except for blur sigma 0.0), there is a lower condition number.

In general, the condition number doesn't change along the y-axis (blur sigma) but only along x-axis (clustering thresholds). This behavior apparently means that COMMIT blur doesn't affect the lowering of the condition number unlike, instead, of the clustering. Removing the number of stream-

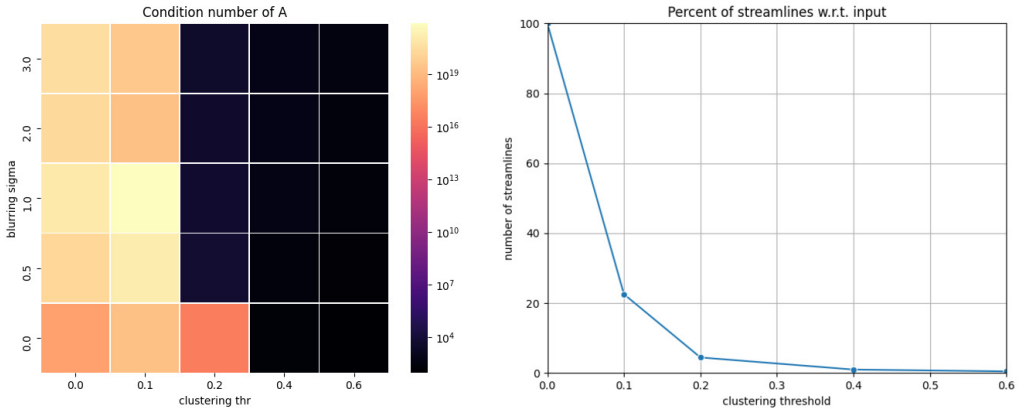


Figure 4.4: Results about condition number experiments.

lines (Figure 4.4), therefore also the collinearity, clustering plays a fundamental role. Indeed, it manages to drastically lower the condition number of the linear system. Note that the problem is, anyway, an ill-conditioned problem since that the condition number is always $\gg 1.0$ for each blurring sigma and each clustering threshold.

Effects of perturbations in the data (y)

In order to investigate and prove the results obtained, reported in the condition number plot (Figure 4.4), I carry out further analysis. Since, by definition, the condition number is how much the output value of the function can change for a small change in the input argument (response to input perturbations of the linear system), I introduce variability in the input data applying low variations of input fitting to the COMMIT system. To do this, I modify the ground truth density map introducing Gaussian noise [100], in this way, I am modifying the $\mathbf{y}$ variable of the linear system (1.6). I create a mask of the density map in order to apply the Gaussian noise only in the region where the voxels values are different from zero and not in the background of the image. Since Gaussian noise has a probability density function which is a normal distribution, I set μ mean noise equal to 1.0 and σ standard deviation noise equal to $\frac{\mu}{SNR}$. Thus, to create a noisy image, I add to the original

image a normal distribution with mean equal to zero and standard deviation which coincides with the standard deviation noise. I repeat this alteration on ground truth map original for ten times, then I execute ten pipeline runs, each with one different alteration of density map respect to the original map. I execute more repetitions because, in this way, my results are more robust and solid. Note that, setting the standard deviation of normal distribution for Gaussian noise, I use SNR concept.

Signal-to-noise ratio (SNR) is defined as the ratio of the power spectral density of a signal with respect to the power of the background noise [101]. The SNR is calculated as follows:

$$SNR(f) = \frac{S(f)}{N(f)} \tag{4.2}$$

where $S(f)$ is the measured signal and $N(f)$ is the measured noise. Obviously, being the noise in the denominator, the more we increase the noise and the more the signal-to-noise ratio decreases. In the experiments, I have used SNR equal to 30 because this value is a good trade-off for the signal to noise ratio.

Therefore, in these experiments, the COMMIT fit was performed keeping the same tractogram and changing the density map to fit to. Once COMMIT is executed, I get the connectomes created after the fit with ten density alterations (10 for COMMIT and 10 for COMMIT blur), then, I can calculate the distances L_1 and L_2 norm between the ground truth density original connectome (without noise) and COMMIT on raw connectome. The same procedure repeated also between the ground truth density original connectome (without noise) and COMMIT blur connectomes. For this experiments, I use an only clustering threshold equal to 0.40mm and use a blur sigma equal to 0.40mm [Table 4.3]. Looking at Figure 4.5, we can deduce that both L_1 COMMIT + blur and L_2 COMMIT + blur distances between connectomes as compared to the original ground truth connectome are always less than that COMMIT on raw, with respect to, always, the original ground truth connectome. This aspect means that the COMMIT + blur system is less conditioned by input variations system compared to COMMIT on raw. Ac-

EXPERIMENTS	
	SD_STREAM
number of streamlines	1 million
tracking_step	0.50mm
tracking_angle	45°
clustering thresholds	0.4
blur sigma	0.4
sampling_step	0.25mm

Table 4.3: Table of SNR experiments.

cordingly, COMMIT blur is more stable and solid with respect to COMMIT on raw.

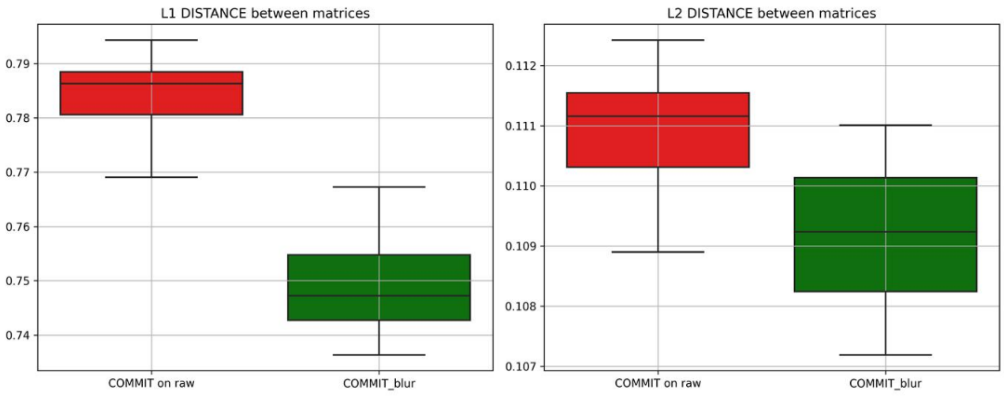


Figure 4.5: Distances between ground truth density map original connectome with COMMIT on raw then with COMMIT + blur connectomes.

Effects of perturbations in the tractogram (A)

In these experiments, I modify the original tractogram passed as input to COMMIT (variable **A** of the linear system (1.6), which is the observation matrix (i.e., dictionary) in every voxel). In this case, I take and modify the original tractogram, in order to create small variations of the original tractogram. To do this, the following strategy is used. Each point (with 3

coordinates P_1, P_2, P_3 because voxel) of each streamline of the original tractogram is shifted of a Δ equal to 0.1mm for a random direction (Figure 4.6).

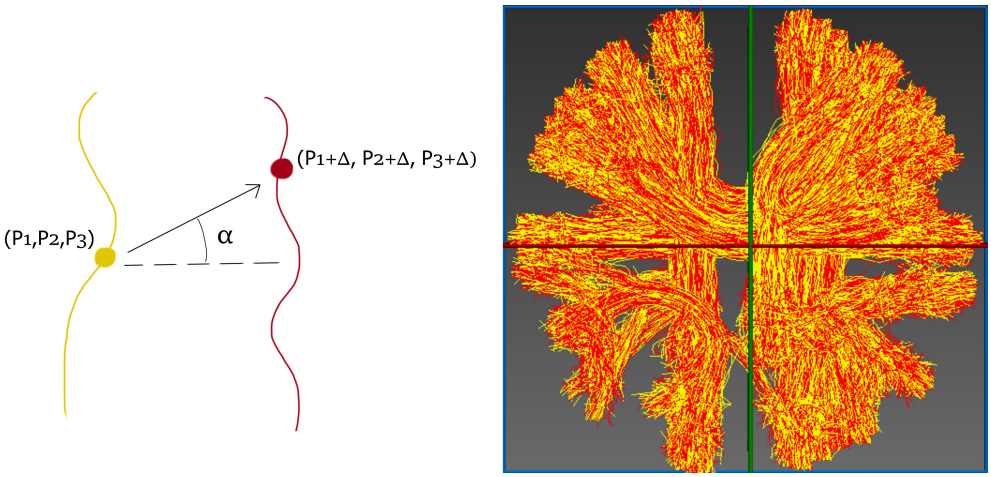


Figure 4.6: Explanation of the concept about shifted points, where the yellow streamline is from original tractogram whereas red streamline is from shifted tractogram (on the left panel). Two differerent tractograms results, the original one is yellow, the shifted one is red (on the right panel). Image obtained with MI-Brain [36]

In Figure 4.6, on the right panel, there is an example of shifted tractogram application. I create ten shifted different variations of the original tractogram in order to execute solid experiments. Now, I create the dictionary COMMIT and COMMIT + blur using both the original tractogram and each shifted tractograms just created.

At this point, once obtained COMMIT and COMMIT + blur results for original tractogram and the shifted tractograms, I can start with analysis. First of all, we need to analyze the respective connectomes. Thus, as first experiment, I compute L_1 norm distances between the COMMIT on raw and COMMIT blur original tractogram connectomes with the ten shifted tractograms connectomes (Figure 4.7). The conclusion we can draw from this is that the L_1 distance trend is lower for COMMIT blur, this means that the latter system reacts better to the perturbations applied on the $\mathbf{A}$ matrix as compared to COMMIT on raw.

Since the connectomes are the direct consequence of the streamline weights,

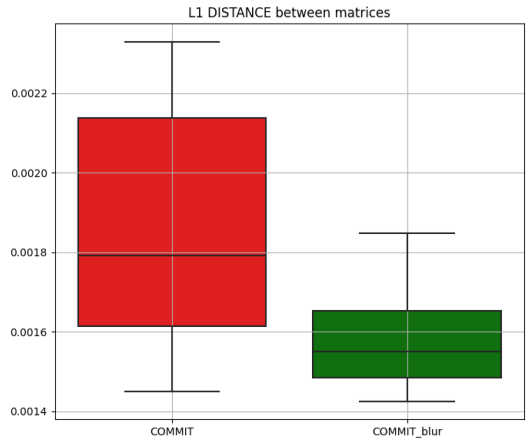


Figure 4.7: L_1 norm distance between original tractogram and shifted tractograms both for COMMIT and for COMMIT blur using connectomes.

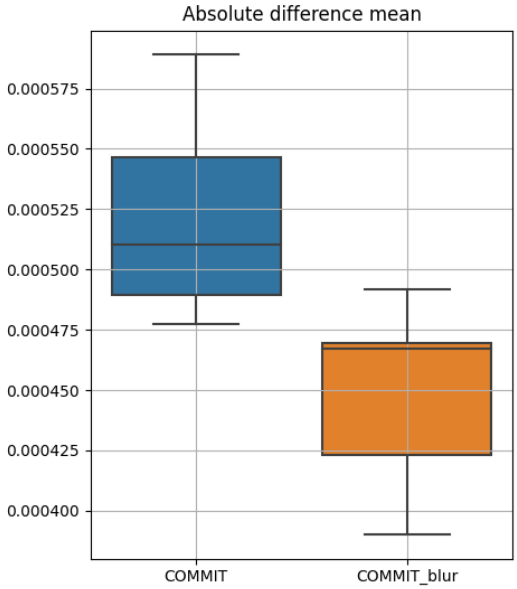


Figure 4.8: Absolute difference mean between original tractogram and shifted tractograms both for COMMIT and for COMMIT blur using streamline weights.

now, I investigate the weight of each streamline of the input tractogram, once applied on it COMMIT and COMMIT blur. To compute this, I have to analyze the the weight of each streamline. After running COMMIT, a vector is created of length as long as the number of streamlines is of the input tractogram and, for each streamline, its weight is obtained. I repeat the same process that I used for the connectomes, but, instead of computing the distances and differences between them, I compute distances and absolute differences between the streamline weights of the original tractogram and the ten streamline weights of the shifted tractograms. Indeed, looking at Figure 4.8, also in this case, as for the connectomes, the COMMIT blur trend is lower than COMMIT on raw, concept that reinforces the results previously obtained from connectomes analysis.

4.2.2 Accuracy of the reconstructions

Valid and invalid bundles

At this point, once terminated the analysis about the condition number, we focus on accuracy of the reconstructions and how accuracy changes according to the blur radii and the clustering thresholds. As seen previously (subsec 4.1.4), to carry out an accuracy study, we use parameters seen for the reconstructions streamlines (VB, IB, RMSE, Time, RAM) [Table 4.1]. As shown in the Figure 4.9, for deterministic data-set we have always the maximum number of VB (27), except with blur radius greater than 4.00mm where we find a loss of one VB for almost all clustering thresholds. Note that, the same behavior occurs when the clustering threshold is greater than 1.00mm. Instead, the trend is different for probabilistic data-set, where we have the maximum number of VB (27) except with blur radius greater than 3.50mm and clustering threshold greater than 1.50mm, in this case we notice that we have a loss of even a couple of VB, therefore worse than the deterministic data-set. This loss is more pronounced in the probabilistic data-set, caused by its random conformation for the streamline created. In these analysis there is the certainty that this loss of VB is linked exclusively to COMMIT, *the cluster does not affect any streamline bundle.*

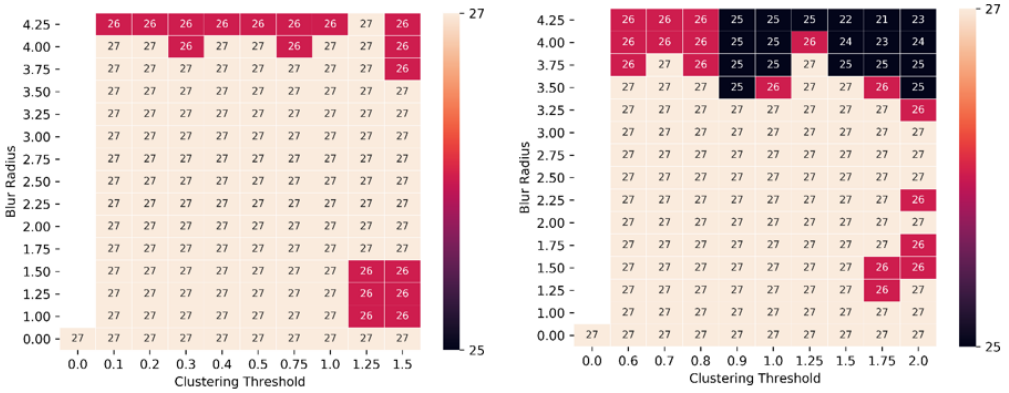


Figure 4.9: Valid bundles experiments. On the left *deterministic* data-set, on the right *probabilistic* data-set. Image taken from [59].

The Figure 4.10 shows the percentage of IB reduction with the use of blurred streamlines after the clustering. Here we can notice that there is a growth gradient [102] that goes from right to left and from bottom to top (diagonal). Thus, the trend is that we drastically reduce the IB using blur radii and clustering thresholds bigger, taking into account that, increasing the size of the blur radius, increases the computational cost and RMSE. Hence, the choice of which measure to choose to optimize the reduction of false positives is a trade-off between performance and error fit. We are able to arrive at a percentage reduction around 70% about IB (*false positives*).

Fitting quality

As shown in Figure 4.11, the blur application decreases RMSE as compared to COMMIT on raw, but, any case, after a certain blur radius, also with blurred streamlines, the error increases linearly. This because, COMMIT isn't able to fit the model in the best possible way, and so we have an higher error.

Computational cost

In Figure 4.12, there is the trend of the time spent (in minutes) during the pipeline. Higher the clustering threshold we apply, lower time spent. This is

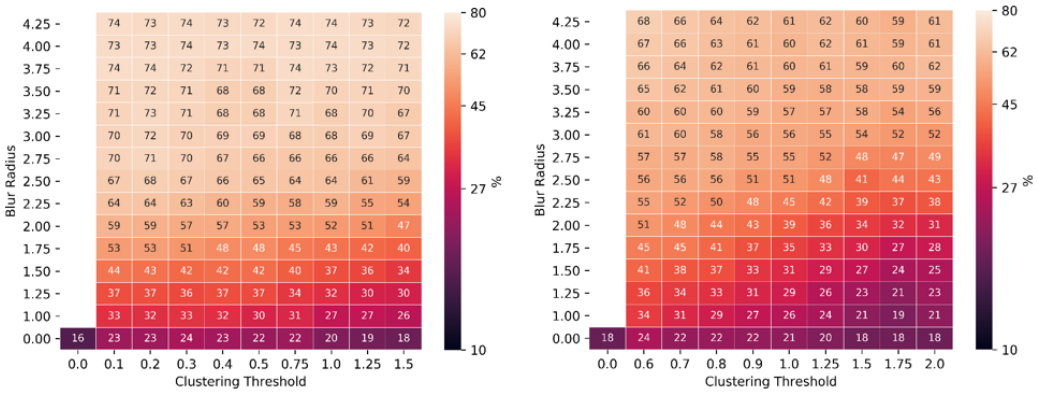


Figure 4.10: Invalid bundles experiments. On the left *deterministic* data-set, on the right *probabilistic* data-set. Image taken from [59].

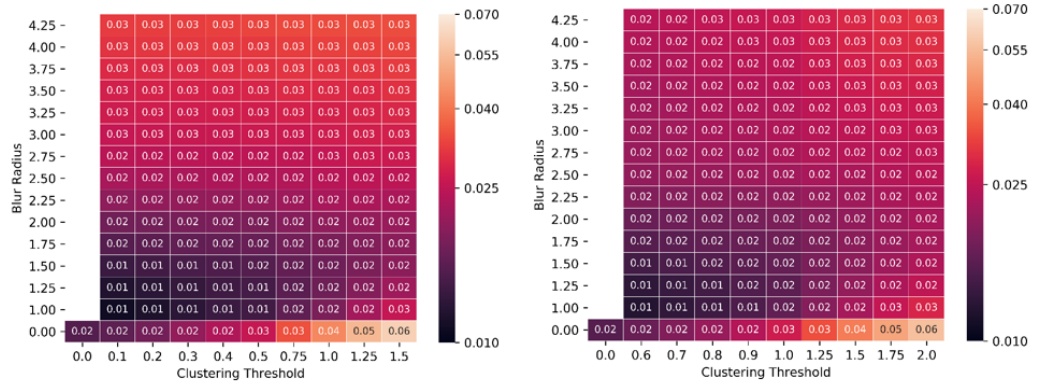


Figure 4.11: RMSE experiments. On the left *deterministic* data-set, on the right *probabilistic* data-set. Image taken from [59].

because at higher thresholds we have less bundles and so the computational time of execution decreased. Whereas, increasing the sigma, and therefore, the radius of the blur streamlines, with the same clustering threshold, the time spent is higher increasing the blur radius, this because it requires more computational time. *This proves how important it is to apply clustering to reduce the size of the data-set and remove the collinearity, to reduce the time required for the analysis.*

Looking at the Figure 4.13, we have the trend of RAM consumption in the framework at its maximum point. We can notice that these two plots

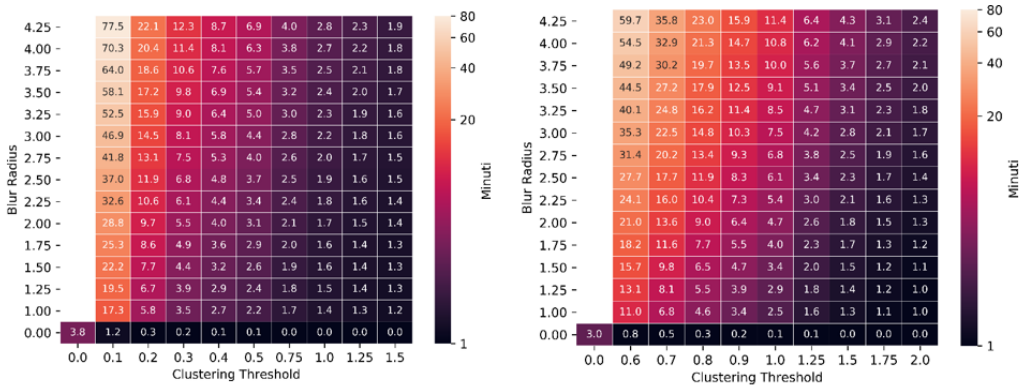


Figure 4.12: Time experiments. On the left *deterministic* data-set, on the right *probabilistic* data-set. Image taken from [59].

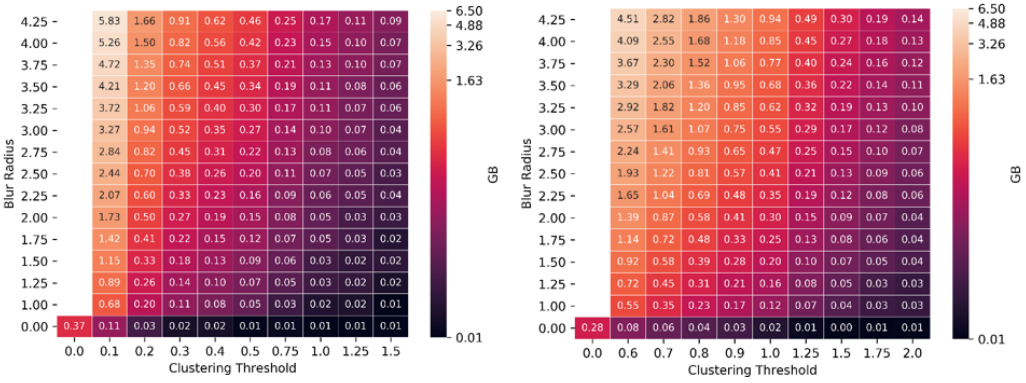


Figure 4.13: RAM experiments. On the left *deterministic* data-set, on the right *probabilistic* data-set. Image taken from [59].

have approximately the same trend as the time plots. RAM consumption is huge when the data-set is dense (low clustering thresholds), instead, RAM is greater when the blur radius increases.

4.2.3 Repeatability

Repeatability is the closeness of the agreement between the results of successive measurements of the same measure, when carried out under the same conditions of measurement [103]. Until now, we have performed analyses to study the accuracy of the method, indeed, we studied the results for one

pipeline run, but, at this point, we have to carry out other analyses to study the *repeatability* of the method and then study if the method is solid and reliable even on several repetitions (Figure 4.14). In order to do this, I have to repeat the analyses executed before on more pipeline runs, for example, 50 runs, in order to be more solid. Thus, I have to execute 50 times the pipeline (*Tractography* $\rightarrow$ *Clustering* $\rightarrow$ *COMMIT* and *COMMIT + blur*). During the repeatability study, we focus on probabilistic algorithm creation, so we will have 50 different probabilistic tractograms. We have chosen this kind of tractography's algorithm because if we had chosen the deterministic algorithm, always keeping the same parameters for repetition, we would have always obtained the same tractogram, by definition.

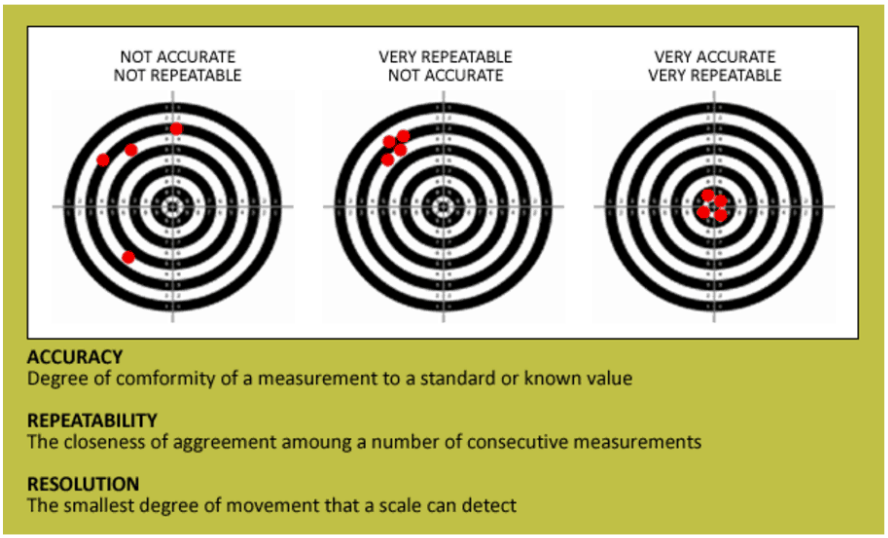


Figure 4.14: Showing accuracy and repeatability concept. Image taken from [104].

50 runs analysis

For the 50 runs I compute the *mean* and the *standard deviation* among all the 50 heatmaps. In each single run I use this data [Table 4.4]. In this way, we will get as result a single heatmap which represents the *mean* and another which represents the *standard deviation* of the respective measured

EXPERIMENTS	
	iFOD2
number of streamlines	1 million
tracking_step	0.50mm
tracking_angle	45°
clustering thresholds	[0.0, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0]
blur radius	[0.0, 1.00, 1.25, 1.50, 1.75, 2.00, 2.25, 2.50, 2.75, 3.00, 3.25, 3.50, 3.75, 4.00]
sampling_step	0.25mm

Table 4.4: Table of repetitions experiments.

value (VB, IB and RMSE). For example, in Figure 4.15, on the left, there is the *valid bundles mean* among all the 50 runs and, on the right, the *valid bundles standard deviation* among all the 50 runs.

In the valid bundles plot (Figure 4.15), increasing both the clustering thresholds and the blur radius, there is a decrease of the *valid bundles mean.* This concept reinforces the theory for which in the top right of the VB mean heatmap too many streamlines have been lost and, doing it this way, we have thinned out too much and the model is no longer representative, as we saw previously for a single run.

In the invalid bundles plot (Figure 4.16) there is, in this case, a better representation of the gradient [102] of the invalid bundles compared to the single run (Figure 4.10). If we move on the diagonal from the lower left corner to the top right corner of the *invalid bundles mean* plot, there is a decrease of the invalid bundles. To highlight how COMMIT + blur drastically reduces the IB number unlike COMMIT on raw (first row 0.00) (Figure 4.16), where instead the IB number is still very high. Applying COMMIT blur, as we have seen before, we reduce IB number, but, at the expense, of a higher computational cost and RMSE. Indeed, a optimal trade-off is, for example, 0.9 clustering threshold and 2.00mm of blur radius, in order to maximize

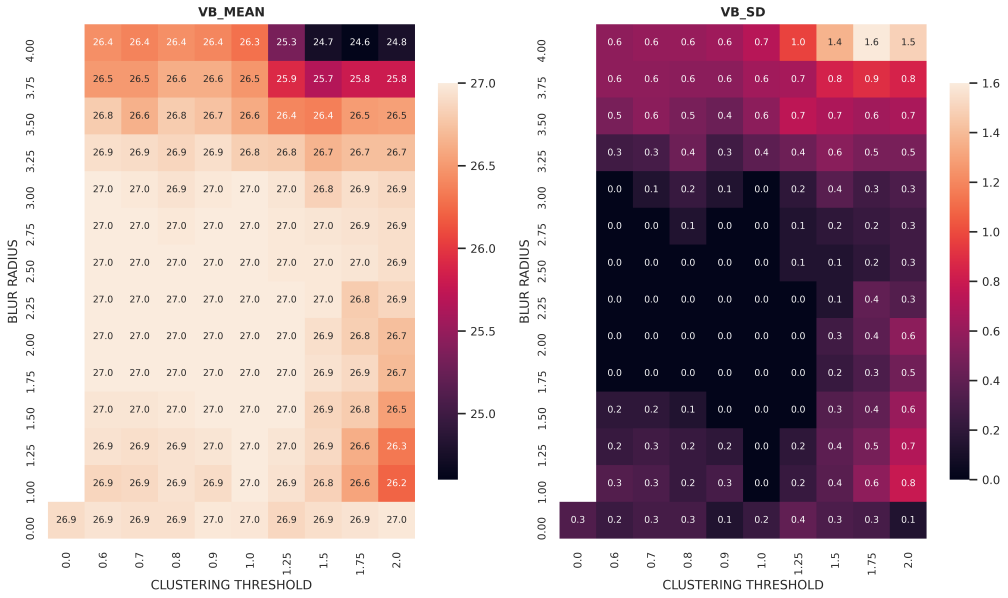


Figure 4.15: Results valid bundles 50 runs, the mean on the left, the standard deviation on the right.

the effectiveness of the clustering filtering and of the COMMIT blur and to minimize RMSE of the fit and computational costs.

In the RMSE plot (Figure 4.17), there is the opposite behavior as compared to the invalid bundles. In fact, there is still a gradient, moving towards the top right corner of the RMSE plot, the RMSE values increment with increasing the blur radius and the clustering thresholds, except the first row (blur radius equal to 0.00), where RMSE values are always very high, unless for the last clustering threshold (2.00mm). This concept confirms the notion that COMMIT blur decreases the fit error as compared to COMMIT on raw for lower blur radii, whereas, higher for blur radii and clustering thresholds RMSE increases linearly.

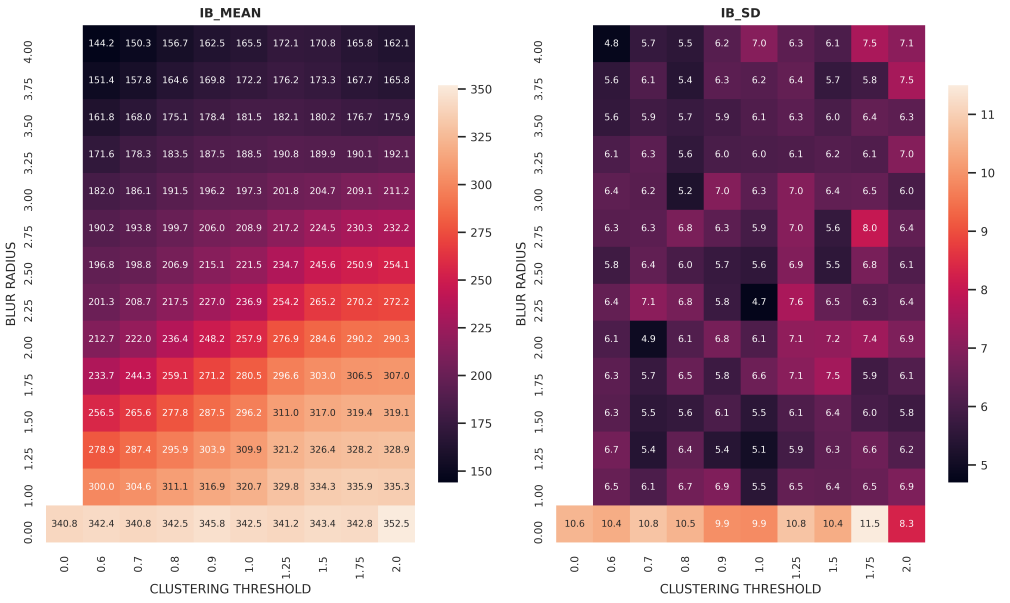


Figure 4.16: Results invalid bundles 50 runs, the mean on the left, the standard deviation on the right.

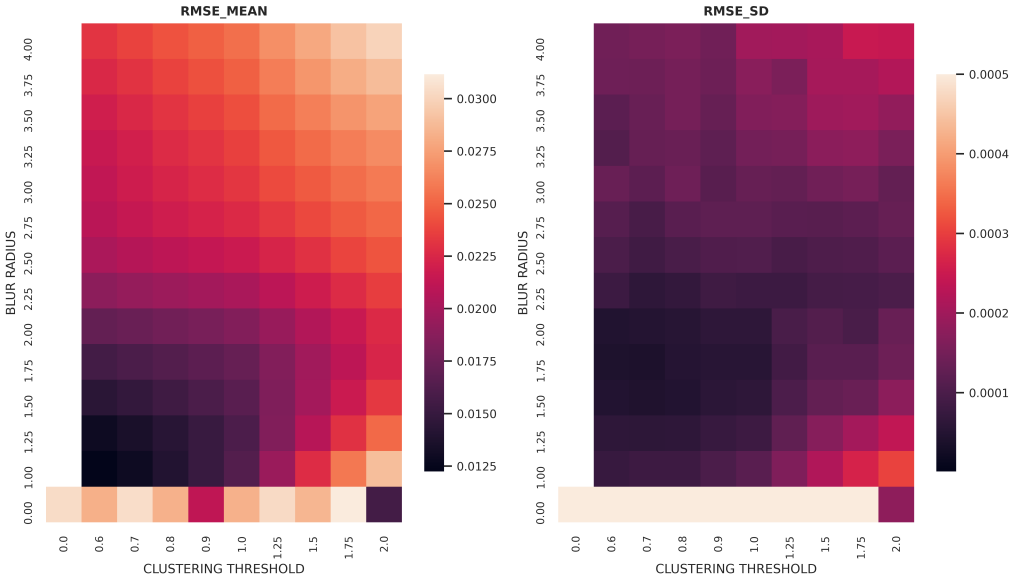


Figure 4.17: Results RMSE 50 runs, the mean on the left, the standard deviation on the right.

Chapter 5

Conclusions and future perspectives

In this thesis, I studied the characterization of a new model, with the introduction of a new concept, the *Blurred Streamline*. The clustering phase is performed to reduce the redundancy problem present in all the tractograms. This reduction removes the collinearity problem in the COMMIT **A** matrix. For clustering, I have used the algorithm for the division of the bundled tractogram, the application of QuickBundles on each of them and the reconstruction of the tractogram. This will not have the problem of collinearity and the execution time will be reduced because the analyzed tractogram is in turn reduced. The creation of the blurred streamlines aims to best describe the signal of the clustered tractogram that has holes.

In particular, I have studied the COMMIT convergence aspect, exploiting the condition number concept to visualize how the COMMIT system reacts to small input changes. After that, I have studied the cause that leads to a lowering of the condition number. Indeed, modifying the data of the linear system ($\mathbf{y}$), I have analyzed this concept through the SNR experiment studying the different connectomes, I drew a conclusion, that there is a reduction of distances and of absolute difference between original ground truth connectivity density map and COMMIT blur respect to COMMIT on raw. Furthermore, I have studied the effects of noise in the tractogram (variable **A**

of the linear system), I have modified and shifted the original tractogram calculating both L_1 distance between connectomes and the absolute difference mean between streamline weights.

Besides, I have also investigated, proved and confirmed the robustness of the method and of the results proposed in previous projects, such as in previous theses [59]. Indeed, I have investigated the accuracy of the reconstructions, in particular using repeatability analysis performing not one run but 50 runs, so that, the results obtained were more stable, detailed and robust respect to the single run results. The conclusions obtained on the repeatability analysis are: as we see for a single run [59], even with 50 runs the trend of the VB indicate that at high cluster thresholds and at high sigma blur a couple of VB tend to be removed (Figure 4.15). As regards, instead, the IB, the trend reflects the analysis carried out on the single run but, in this case, there is a more evident gradient than before, this indicates a greater robustness and solidity of the results. For the IB (Figure 4.16), increasing the sigma blur, there is a reduction of the IB and that, the lower the clustering threshold and the lower the IB at the level of the same sigma blur, at the expense of more time and space computation (Figure 4.12) (Figure 4.13). RMSE trend, instead, is the opposite to IB trend, it increases in a linear way whether we increase both blur sigma and the clustering threshold, except for blur sigma equal to zero (COMMIT on raw) where RMSE is however high (Figure 4.17). Therefore, the results suggest us that ideally it would be enough to increase both the blur radius and the clustering thresholds, but this, at the expense of a greater consumption of resources and computational time.

Other analysis could be carried out in the future, for example:

- focusing on other parameters, for a deeper knowledge of the method, on which I did not dwell on in detail in this thesis, for example, the sampling step and the numbers of sectors (*blurred streamlines*) and how to modify them for better fit and performance;

- application of FFClust algorithm in the pipeline to observe if there are improvements in tractogram results and performance compared to the

other clustering algorithms we have applied so far;

- experiments performed in vivo data instead of synthetic data;
- a study and characterization on the parameters and behavior of the COMMIT upgrade, COMMIT2 [105].

Bibliography

[1] Britannica. Human nervous system.

[2] Verywellmind. How the peripheral nervous system works.

[3] Christopherreeve. How-the-spinal-cord-works.

[4] Wang Jae Lee. *Central Nervous System (CNS)*, pages 101–117. Springer Netherlands, Dordrecht, 2019.

[5] Wikipedia. Neurons — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].

[6] Britannica. Cerebrospinal-fluid.

[7] Isabelle Dussauge. Technomedical visions : Magnetic resonance imaging in 1980s sweden. 01 2008.

[8] Zipursky SL Lodish H, Berk A. *Molecular Cell Biology. 4th edition.* 2000.

[9] Technologynetworks. Gray-matter-vs-white-matter.

[10] Thijs Dhollander. *Accounting for Complex Structure in Diffusion Weighted Imaging Data using Volume Fraction Representations.* PhD thesis, 04 2014.

[11] Sciencebeta. Grey matter definition.

[12] C. Sampaio-Baptista and H. Johansen-Berg. White Matter Plasticity in the Adult Brain. *Neuron*, 96(6):1239–1251, 12 2017.

[13] BrainFacts. Myelin: An overview.

[14] A.Daducci. Slides daducci's course biomedical image processing, 2020.

[15] BioMiamiEdu. Brainanalogy.

[16] Exactradiology. risonanza magnetica.

[17] Michael Jacobs. Totally accessible mri: A user's guide to principles, technology, and applications. *Medical Physics*, 35(11):5198–5198, 2008.

[18] Nibib. Magnetic resonance imaging (mri).

[19] Kathryn Broadhouse. The physics of mri and how we use it to reveal the mysteries of the mind. *Frontiers for Young Minds*, 7, 03 2019.

[20] Heather Haynes and William Holmes. *The Emergence of Magnetic Resonance Imaging (MRI) for 3D Analysis of Sediment Beds*. 12 2013.

[21] Nih. Nibib, 2011.

[22] Olivier Keunen, Torfinn Taxt, Renate Grüner, Morten Lund-Johansen, Joerg Tonn, Tina Pavlin, Rolf Bjerkvig, Simone Niclou, and Frits Thorsen. Multimodal imaging of gliomas in the context of evolving cellular and molecular therapies. *Advanced Drug Delivery Reviews*, 76, 09 2014.

[23] Robert Le, Minh Nguyen, and Weiqi Yan. *A Web-Based Augmented Reality Approach to Instantly View and Display 4D Medical Images*, pages 691–704. 02 2020.

[24] Wikipedia. brownian — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].

[25] Adolf Fick. Ueber diffusion. *Annalen der Physik*, 170(1):59–86, 1855.

[26] D.K. Jones. *Diffusion MRI: Theory, Methods, and Applications*. Oxford University Press, 2010.

[27] Denis Le Bihan and Mami Iima. Diffusion magnetic resonance imaging: What water tells us about biological tissues. *PLOS Biology*, 13(7):1–13, 07 2015.

[28] Alessandro Stecco and Alfonso Ragozzino. *L'ESSENZIALE... NELL'IMAGING RM DI DIFFUSIONE*. Public Library of Science.

[29] Siân Price. Multiple sclerosis: diagnostic issues and modern management. *British and Irish Orthoptic Journal*, 6:5, 08 2009.

[30] Jonathan H. Burdette, David D. Durden, Allen D. Elster, and Yi-Fen Yen. High b-value diffusion-weighted mri of normal brain. *Journal of Computer Assisted Tomography*, 2001.

[31] Neetu Soni, Anant Mehrotra, Sanjay Behari, Sunil Kumar, and Nishant Gupta. Diffusion-tensor imaging and tractography application in preoperative planning of intra-axial brain lesions. *Cureus*, 9:e1739, 03 2017.

[32] Denis Le Bihan, Jean-François Mangin, Cyril Poupon, Chris A. Clark, Sabina Pappata, Nicolas Molko, and Hughes Chabriat. Diffusion tensor imaging: Concepts and applications. *Journal of Magnetic Resonance Imaging*, 13(4):534–546, 2001.

[33] Yu-li You, Mostafa Kaveh, Wenyuan Xu, and Allen Tannenbaum. Analysis and design of anisotropic diffusion for image processing. pages 497–501, 01 1994.

[34] P. Mukherjee, J.I. Berman, S.W. Chung, C.P. Hess, and R.G. Henry. Diffusion tensor mr imaging and fiber tractography: Theoretic underpinnings. *American Journal of Neuroradiology*, 29(4):632–641, 2008.

[35] Ben Jeurissen, Maxime Descoteaux, Susumu Mori, and Alexander Leemans. Diffusion mri fiber tractography of the brain. *NMR in Biomedicine*, 32(4):e3785, 2019. e3785 NBM-17-0045.R2.

[36] Imeka. Mibrain.

[37] O. Ciccarelli, T. E. Behrens, D. R. Altmann, R. W. Orrell, R. S. Howard, H. Johansen-Berg, D. H. Miller, P. M. Matthews, and A. J. Thompson. Probabilistic diffusion tractography: a potential tool to assess the rate of disease progression in amyotrophic lateral sclerosis. *Brain*, 129(7):1859–1871, 05 2006.

[38] Leon Stefanovski, Paul Triebkorn, Andreas Spiegler, Margarita-Arimatea Diaz-Cortes, Ana Solodkin, Viktor Jirsa, Anthony McIntosh, and Petra Ritter. Linking molecular pathways and large-scale computational modeling to assess candidate disease mechanisms and pharmacodynamics in alzheimer's disease. *Frontiers in Computational Neuroscience*, 13, 08 2019.

[39] Simona Schiavi, Mario Ocampo-Pineda, Muhamed Barakovic, Laurent Petit, Maxime Descoteaux, Jean-Philippe Thiran, and Alessandro Daducci. A new method for accurate in vivo mapping of human brain connections using microstructural and anatomical information. *Science Advances*, 6(31), 2020.

[40] Wikipedia. Roc curve — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].

[41] Pulmonarychronicles. The receiver operating characteristic (roc) curve.

[42] Shengping Yang and Gilbert Berdine. The receiver operating characteristic (roc) curve. *The Southwest Respiratory and Critical Care Chronicles*, 5:34, 05 2017.

[43] Alessandro Daducci, Alessandro Dal Palú, Maxime Descoteaux, and Jean-Philippe Thiran. Microstructure informed tractography: Pitfalls and open challenges. *Frontiers in Neuroscience*, 10:247, 2016.

[44] Derek K Jones, Thomas R Knösche, and Robert Turner. White matter integrity, fiber count, and other fallacies: the do's and don'ts of diffusion mri. *Neuroimage*, 73:239–254, 2013.

[45] Alessandro Daducci, Alessandro Palú, Alia Lemkaddem, and Jean-Philippe Thiran. Commit: Convex optimization modeling for microstructure informed tractography. *IEEE Transactions on Medical Imaging*, 12 2014.

[46] Moises Hernandez-Fernandez, Istvan Reguly, Saad Jbabdi, Mike Giles, Stephen Smith, and Stamatios N. Sotiropoulos. Using gpus to accelerate computational diffusion mri: From microstructure estimation to tractography and connectomes. *NeuroImage*, 188:598–615, 2019.

[47] Alessandro Daducci. COMMIT. https://github.com/daducci/COMMIT, 2015.

[48] Anthony J. Sherbondy, Robert F. Dougherty, Rajagopal Ananthanarayanan, Dharmendra S. Modha, and Brian A. Wandell. Think global, act local; projectome estimation with bluematter. In Guang-Zhong Yang, David Hawkes, Daniel Rueckert, Alison Noble, and Chris Taylor, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, pages 861–868, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[49] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[50] Iman Aganj, G. Sapiro, and Noam Harel. Q-space modeling in diffusion-weighted mri. *Brain Mapping: An Encyclopedic Reference*, 1:257–263, 12 2015.

[51] Wikipedia. Condition number.

[52] Habshah Midi, Saroje Sarkar, and Sohel Rana. Collinearity diagnostics of binary logistic regression model. *Journal of Interdisciplinary Mathematics*, 13:253–267, 05 2013.

[53] Alessandro Daducci, Francesco Gobbi, Nicola Febbrari, Matteo Battocchio, and Simona Schiavi. Blurred streamlines: a new concept to improve tractography accuracy by spatially blurring signal contributions.

In *Proceedings of the International Society for Magnetic Resonance in Medicine (ISMRM)*, 2021.

[54] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. *Singular Value Decomposition and Principal Component Analysis*, pages 91–109. Springer US, Boston, MA, 2003.

[55] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[56] Towardsdatascience. Simple svd algorithms.

[57] F. Frenet. Sur les courbes à double courbure. *Journal de Mathématiques Pures et Appliquées*, pages 437–447, 1852.

[58] J.-A. Serret. Sur quelques formules relatives à la théorie des courbes à double courbure. *Journal de Mathématiques Pures et Appliquées*, pages 193–207, 1851.

[59] Francesco Gobbi. Analisi e caratterizzazione di un nuovo modello di rappresentazione della connettività cerebrale. Master's thesis, Verona, 2021.

[60] Tong Luo, Huan Chen, and Ghassan Kassab. Resliced image space construction for coronary artery collagen fibers. *PLOS ONE*, 12:e0184972, 09 2017.

[61] T. Soni Madhulatha. An overview on clustering methods, 2012.

[62] Geeksforgeeks. Clustering in machine learning.

[63] Towardsdatascience. Hierarchical clustering.

[64] Mahamed Omran, Andries Engelbrecht, and Ayed Salman. An overview of clustering methods. *Intell. Data Anal.*, 11:583–605, 11 2007.

[65] Chen Fu and Jianhua Yang. Granular classification for imbalanced datasets: A minkowski distance-based method. *Algorithms*, 14:54, 02 2021.

[66] Jiachi Zhang, Liu Liu, Yuanyuan Fan, Lingfan Zhuang, Tao Zhou, and Zheyan Piao. Wireless channel propagation scenarios identification: A perspective of machine learning. *IEEE Access*, PP:1–1, 03 2020.

[67] Wikipedia. K-means — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].

[68] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2007.

[69] Yu-Zhong Chen and Ying-Cheng Lai. Universal structural estimator and dynamics approximator for complex networks. 11 2016.

[70] Rachad Atat, Lingjia Liu, Jinsong Wu, Guangyu Li, Chunxuan Ye, and Yang Yi. Big data meet cyber-physical systems: A panoramic survey. *IEEE Access*, 6:73603–73636, 11 2018.

[71] Wikipedia. hierachical, 2011.

[72] Vadym Mozgovoy. Exploration of scenario-based simulations for stress benchmarking in swiss public service. In *Exploration of Scenario-based Simulations for Stress Benchmarking in Swiss Public Service*, pages 23–28, 06 2020.

[73] Sudha V and Girijamma H a. Appraising research direction & effectiveness of existing clustering algorithm for medical data. *International Journal of Advanced Computer Science and Applications*, 8, 03 2017.

[74] Towardsdatascience. Fuzzy clustering.

[75] R. Manikandan, Ambeshwar Kumar, and Deepak Gupta. Chapter 5 - hybrid computational intelligence for healthcare and disease diagnosis. In Siddhartha Bhattacharyya, Václav Snášel, Deepak Gupta, and Ashish Khanna, editors, *Hybrid Computational Intelligence*, Hybrid Computational Intelligence for Pattern Analysis and Understanding, pages 97–122. Academic Press, 2020.

[76] Eleftherios Garyfallidis, Matthew Brett, Marta Correia, Guy Williams, and Ian Nimmo-Smith. Quickbundles, a method for tractography simplification. *Frontiers in Neuroscience*, 6:175, 2012.

[77] Sven Kosub. A note on the triangle inequality for the jaccard distance, 2016.

[78] Eleftherios Garyfallidis, Matthew Brett, Bagrat Amirbekian, Ariel Rokem, Stefan Van Der Walt, Maxime Descoteaux, and Ian Nimmo-Smith. Dipy, a library for the analysis of diffusion mri data. *Frontiers in Neuroinformatics*, 8:8, 2014.

[79] Garyfallidis Eleftherios, Côté Marc-Alexandre, Rheault François, and Descoteaux Maxime. Quickbundlesx: Sequential clustering of millions of streamlines in multiple levels of detail at record execution time. In *ISMRM 27th Annual Meeting & Exhibition*, number POST_TALK in prova, 2016.

[80] Andrea Vázquez, Narciso López-López, Alexis Sánchez, Josselin Houenou, Cyril Poupon, Jean-François Mangin, Cecilia Hernández, and Pamela Guevara. Ffclust: Fast fiber clustering for large tractography datasets for a detailed study of brain connectivity. *NeuroImage*, 220:117070, 2020.

[81] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[82] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 1177–1178, New York, NY, USA, 2010. Association for Computing Machinery.

[83] Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya, and Muljono. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 533–538, 2018.

[84] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.

[85] Mango. Mango tool.

[86] J-Donald Tournier, Robert Smith, David Raffelt, Rami Tabbara, Thijs Dhollander, Maximilian Pietsch, Daan Christiaens, Ben Jeurissen, Chun-Hung Yeh, and Alan Connelly. Mrtrix3: A fast, flexible and open software framework for medical image processing and visualisation. *NeuroImage*, 202:116137, 2019.

[87] MRtrix. tckgen explanation.

[88] Flavio Dell'Acqua and J.-Donald Tournier. Modelling white matter with spherical deconvolution: How and why? *NMR in Biomedicine*, 32(4):e3945, 2019. e3945 NBM-17-0226.R1.

[89] MRtrix. tck2connectome explanation.

[90] MRtrix. connectome2tck explanation.

[91] Towardsdatascience. Rmse definition.

[92] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[93] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[94] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[95] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[96] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[97] Matthew Brett, Christopher J. Markiewicz, Michael Hanke, Marc-Alexandre Côté, Ben Cipollini, Paul McCarthy, Dorota Jarecka, Christopher P. Cheng, Yaroslav O. Halchenko, Michiel Cottaar, Eric Larson, Satrajit Ghosh, Demian Wassermann, Stephan Gerhard, Gregory R. Lee, Hao-Ting Wang, Erik Kastman, Jakub Kaczmarzyk, Roberto Guidotti, Or Duek, Jonathan Daniel, Ariel Rokem, Cindee Madison, Brendan Moloney, Félix C. Morency, Mathias Goncalves, Ross Markello, Cameron Riddell, Christopher Burns, Jarrod Millman, Alexandre Gramfort, Jaakko Leppäkangas, Anibal Sólon, Jasper J.F. van den Bosch, Robert D. Vincent, Henry Braun, Krish Subramaniam, Krzysztof J. Gorgolewski, Pradeep Reddy Raamana, Julian Klug, B. Nolan Nichols, Eric M. Baker, Soichi Hayashi, Basile Pinsard, Christian Haselgrove, Mark Hymers, Oscar Esteban, Serge Koudoro,

Fernando Pérez-García, Nikolaas N. Oosterhof, Bago Amirbekian, Ian Nimmo-Smith, Ly Nguyen, Samir Reddigari, Samuel St-Jean, Egor Panfilov, Eleftherios Garyfallidis, Gael Varoquaux, Jon Haitz Legarreta, Kevin S. Hahn, Oliver P. Hinds, Bennet Fauber, Jean-Baptiste Poline, Jon Stutters, Kesshi Jordan, Matthew Cieslak, Miguel Estevan Moreno, Valentin Haenel, Yannick Schwartz, Zvi Baratz, Benjamin C Darwin, Bertrand Thirion, Carl Gauthier, Dimitri Papadopoulos Orfanos, Igor Solovey, Ivan Gonzalez, Jath Palasubramaniam, Justin Lecher, Katrin Leinweber, Konstantinos Raktivan, Markéta Calábková, Peter Fischer, Philippe Gervais, Syam Gadde, Thomas Ballinger, Thomas Roos, Venkateswara Reddy Reddam, and freec84. nipy/nibabel: 3.2.1, November 2020.

[98] Charles Guille-Escuret, Baptiste Goujaud, Manuela Girotti, and Ioannis Mitliagkas. A study of condition numbers for first-order optimization, 2020.

[99] Heatmap. Heatmap - definition.

[100] Wikipedia. Gaussian distribution — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].

[101] Alex Suarez-Perez, Gemma Gabriel, Beatriz Rebollo, Xavi Illa, Anton Guimerà-Brunet, Javier Hernández-Ferrer, Maria Teresa Martínez, Rosa Villa, and Maria V. Sanchez-Vives. Quantification of signal-to-noise ratio in cerebral cortex recordings using flexible meas with co-localized platinum black, carbon nanotubes, and gold electrodes. *Frontiers in Neuroscience*, 12:862, 2018.

[102] Wikipedia. Gradient definition.

[103] Wikipedia. Repeatability definition.

[104] Cross. repeatability.

[105] Simona Schiavi, Mario Ocampo-Pineda, Muhamed Barakovič, Laurent Petit, Maxime Descoteaux, Jean-Philippe Thiran, and Alessandro Da-

ducci. A new method for accurate in vivo mapping of human brain connections using microstructural and anatomical information. *Science Advances*, 6:eaba8245, 07 2020.