Hochschule Osnabrück

University of Applied Sciences

Fakultät

Ingenieurwissenschaften und Informatik

Bachelorarbeit

über das Thema:

Konzeption und Realisation eines Visualisierungsmodells zur Darstellung funktioneller Konnektivität im menschlichen Gehirn

Autor:	Lennart Bramlage, 627599
	lennart.bramlage@hs-osnabrueck.de
1. Prüfer:	Herr Prof. Johannes Nehls
2. Prüfer:	Herr Dr. Raffi Enficiaud
Abgabedatum:	01.11.2017

HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Themenblatt

Ausgabetermin des Themas für die Abschlussarbeit: 12.09.17

Für den Studierenden:

Name: Bramlage **Vorname:** Lennart

Matrikel-Nr.: 627599

Studiengang: Media & Interaction Design

Wird folgendes Thema gestellt:

Konzeption und Realisation eines Visualisierungsmodells zur Darstellung funktioneller Netzwerke im menschlichen Gehirn

Erstprüfer(in): Herr Prof. Johannes Nehls

Zweitprüfer(in): Herr Dr. Raffi Enficiaud

Bitte urschriftlich zurück an die

Studierendenverwaltung

Das Thema der Abschlussarbeit wurde ausgegeben am: 12.09.17

Erklärung des Studierenden:

Ich habe zur Kenntnis genommen, dass die Abschlussarbeit in schriftlicher und elektronischer Form spätestens abzugeben ist am: **05.12.17**

Es ist mit den Prüfenden abzustimmen, in welcher Form und auf welchem Weg die Abgabe der elektronischen Fassung der Abschlussarbeit erfolgt. Schriftliche und elektronische Form müssen übereinstimmen.

[illegible]

Unterschrift des Prüflings

Kurzfassung

Seit den frühen Neunzigern haben sich die Methoden zur Ableitung funktioneller Konnektivität stetig weiterentwickelt und entsprechende Datensätze sind feiner aufgelöst als je zuvor. Entsprechende Präsentationsmethoden jedoch sind um einiges weniger weit entwickelt. Schon ein einzelnes funktionelles Netzwerk besteht aus hunderttausenden relevanten Verbindungen was in der Visualisierung einen Kompromiss zwischen Vollständigkeit und Lesbarkeit provoziert. Besonders bei der Visualisierung von Konnektivität mit direkter Referenz zur anatomischen Repräsentation des Gehirns sorgt die schiere Anzahl von Verbindungen für ein undurchdringliches Chaos und verbietet jede Identifikation übergeordneter Tendenzen des Netzwerks.
Innerhalb der letzten Jahre haben Forscher wiederholt Methoden der Cluster Analyse (mean-shift, spectral clustering) verwendet um solche Tendenzen zu extrahieren. Häufig jedoch, wurden die Resultate ohne Bezug zum vollständigen funktionellen Netzwerk visualisiert. In diesem Projekt soll K-Means, als Methode der Cluster Analyse, als Basis für einen Edge-Bundling Ansatz genutzt werden um vollständige, funktionelle Netzwerke lesbar abzubilden. Zu diesem Zweck wurde zunächst der K-Means Algorithmus für sechs-dimensionale Datenpunkte, also die Koordinaten zweier Punkte im 3D Raum, angepasst um damit auf Datensätze funktioneller Konnektivität angewendet werden zu können. Die Resultate wurden in ein, mit Paraview entwickeltes, Visualisierungsmodell integriert. Der realisierte Software Prototyp erlaubt somit die Visualisierung funktioneller Konnektivität als gebündelte Einzelverbindungen, innerhalb oder außerhalb eines Referenzmodells des Gehirns. K-Means in Kombination mit einer Edge-Bundling Lösung erzielte dabei eine erhebliche Klärung der Verbindungsstruktur.

Abstract

Methods of aggregating functional connectivity data have evolved at a fast pace since the early nineties and datasets of functional connectivity can be gathered at higher resolutions than ever. Presentation methods for this data, however, are much less well developed. Even a single subject experiment can yield hundreds of thousands of relevant connections, forcing presentation methods to compromise between completeness and readability. Especially when visualising connectivity data in immediate reference to the anatomical structure of the brain, the sheer number of connections obfuscates the network's overarching tendencies and renders the visualisation cluttered and chaotic.
In recent years, researchers have applied a number of cluster analysis methods (mean-shift, spectral clustering etc.) to identify major network tendencies in functional connectivity data. The results were commonly visualised without regard to the full functional network. This project employs K-Means as a method of cluster analysis and uses it as a basis for edge bundling in a whole-scale network visualisation in order to preserve the entirety of information provided by a benchmark dataset. For this purpose a variant of K-Means was implemented, that can be applied to functional connectivity datasets represented as a set of six-dimensional datapoints, where each datapoint consists of two 3D point coordinates. The results were incorporated into a visualisation model, developed for the VTK application Paraview. The implemented software prototype allows the visualisation of functional connectivity data on the internal or external of the brain as a set of curves in Paraview. K-Means in combination with a bundling method has achieved a significant reduction of screen-space clutter for this visualisation.

Contents

List of Figures

Glossary and Abbreviations

Abbreviations

BOLD
Blood-oxygen-leveldependant

MRI/fMRI
Magnetic resonance imaging, functional magnetic resonance imaging

VTK
Visualization Toolkit

ROI
Region of interest. Describes a certain brain area or fragment thereof.

DWI
Diffusion weighted imaging.

Glossary

Cluster
A group or set of entities sharing similar attributes.

(Interpreted) Interface Layer
Exposure of functionalities implemented in another programming language to the programming language of the current environment.

Functional Connectivity
A type of connectivity, which is not necessarily based on physiological structure. Instead it is a statistical concept, informed by time series correlation or covariance of spatially segregated brain areas.

Connectome
Statistically calculated network of functional or structural connectivity, depending on the scope of the study defining it.

Vertex
A three-dimensional point in space.

Barycentre
Describes the centre of mass of a given set of elements.

1. Introduction

1.1 Overview

Mapping the human brain is a concurrent topic in neuroscience. Researchers hope to infer understanding of the brain's functionality from a well-defined and complete map of its inner workings. This map is generally referred to as the brain's wiring diagram or the 'Connectome', however there is a sensible distinction between the 'structural' and the 'functional Connectome'. The structural (also anatomical) Connectome describes a map of actual, physiological connections between neurons in the brain.
The functional Connectome, on the other hand, describes the interconnectedness of active brain regions during or in the absence of task performance.

Visualisation models are a major element in Connectome research. Plain connectivity data is dense and seldom human readable. Understanding the interconnectedness of brain areas is best achieved with an anatomical reference. Where the visualisation of the structural Connectome is pre-defined by the actual presence of the axons in the white matter regions of the brain, functional Connectivity is only a statistical concept and therefore its visualisation is possible in a number of ways. A common analogy is the mapping of a computers circuitboard, compared to the mapping of a programs execution.

Neither the structural, nor the functional Connectome have been mapped in their entirety. For the structural Connectome, this is largely due to the limitations of current technology in medical imaging. Non-invasive methods like DWI (Diffusion-Tensor-Imaging or DW-MRI Diffusion-Weighted Magnetic Resonance Imaging) and fMRI (functional Magnetic Resonance Imaging) have improved significantly in recent years, but their resolution still only allows the analysis with a level of uncertainty.
Functional Connectivity research is impacted by the same demarcations, however the more prevalent problem is the definition of a desired visualisation model.

Datasets of functional Connectivity hold an immense number of correlating or covarying brain regions, making an accurate but insightful visualisation hard to generate. The most common approach is to limit the areas of interest during the gathering of the data, prior to the visualisation, and concentrating solely on their relations to other areas of the brain. Usually this data is gathered using fMRI while the subject executes a number of simple tasks, or is completely at rest (Resting-State fMRI).

The benchmark dataset in this project however aims to capture a whole-scale functional network, without simplification or summary. This adds new requirements to a visualisation model, namely the display of hundreds of thousands of connections in a single screen. This thesis aims to prototype such a visualisation model.

1.2 Goals

The main goal of this thesis is to create a prototype of a visualisation software, that can display whole-scale, high-field functional connectivity networks. This is supposed to be achieved by applying methods of cluster analysis to functional connectivity data as provided by Dr. Johannes Stelzer and his group at the Max-Planck-Institute for Biological Cybernetics.

Connectivity data will be displayed as a set of curves, spanning areas of previously inferred functional connectivity in 3D-space. A custom K-Means implementation will be developed and will provide a method of defining low-variance clusters among the curves. Results of the clustering will be incorporated into the visualisation by forcing the curves to converge with their respective cluster centroid.

In order to further the discussion on the visualisation of functional connectivity data, the visualisation model proposed in this thesis will attempt to externalise the display of connective strands. As a reference, a 3D mesh of a subject brain will be integrated into the visualisation. The resulting visualisation will show similarities to a magnetic field diagram.

1.3 Structure of this Thesis

The central part of this thesis will be a prototypical software package, that holds all necessary tools to create visualisations of high-resolution functional connectivity.

First, the background chapter will relate the importance of functional connectivity research, general approaches and previous research in the field, that directly informs the course of this thesis. Additionally it will include a description of the experiment in which the benchmark dataset has been gathered and investigate the layout of the dataset, file by file. With this foundation, a set of requirements to the software will be provided, which can be used for reference to later determine the progress of the software project and the outlook on how to proceed after the conclusion of this thesis.

The following chapters will closely examine the implemented methods used in this particular software package, beginning with the drawing of connected edges in 3D space. This will be followed by the two major implementations in the visualisation model; the application of the clustering algorithm K-Means on a dataset of functional connectivity and the incorporation of its results in the visualisation.

After the exploration of the applications logic, another chapter will provide information on the general workflow and usage of the provided software.

2. Background

2.1 General Terms

2.1.1 Functional Magnetic Resonance Imaging (fMRI)

fMRI (functional Magnetic Resonance Imaging) is a non-invasive imaging technique, commonly applied in Cognitive Science and Neuropsychology. The method capitalises on the distinct magnetic properties of oxygenated (i.e. arterial) and deoxygenated (i.e. venous) blood [Ogawa1990] as well as activity-dependant blood flow in the brain (also known as hemodynamics) in order to identify localised neural activity [Singleton2009].

Due to a lack of locally stored energy, activated neurons require glucose and oxygen sourced through hemoglobin [Nieuwenhuys1998]. During activation of a particular brain region, blood vessels expand and blood flow increases resulting in higher concentrations of oxygen-rich blood [Roy1890]. The exact physiology of this process escapes the scope of this thesis. Essentially, the interaction of the MRI induced magnetic field and the fluctuating magnetic properties of blood flow inside the brain provoke a measurable signal termed BOLD-Contrast (or Blood-Oxygen-Level-Dependant) **Fig.2.1** [Ogawa1990].

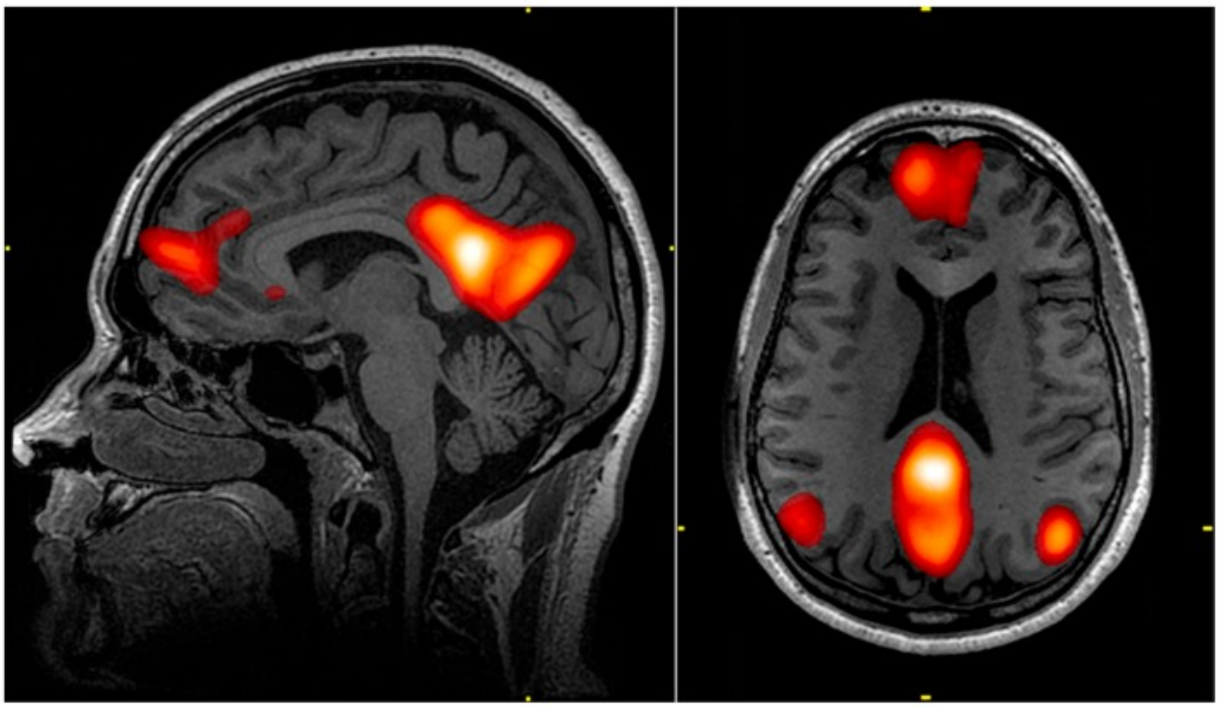


Fig.2.1: highlighted, statistically relevant brain regions in an fMRI scan. BOLD-contrast images are commonly recorded at lower visual resolution and later superimposed on high resolution scans of the subject.

The non-invasive nature and the possibility of Depth imaging have made fMRI the prime technique for researching the human Connectome [Smith2013]. Three major applications of MRI have acquired popularity in Connectome research. While Resting State (or Default-Mode Network) fMRI and Task-Activated fMRI employ the same general technique with a variation only in the experiment design, Diffusion-Weighted Imaging (DWI) is commonly used to map white matter structures as opposed to neural activity, which occurs exclusively in the grey matter. DWI has been used to great effect in mapping the anatomical structures of the brains white matter, furthering research of the structural Connectome [Dillow2010]. However, Resting-State and Task-Related fMRI are more relevant to this thesis, Task-Related fMRI being the method employed to gather the data that is to be visualised.

In Task-Related fMRI the patient or subject are instructed to perform a simple assignment like tapping their fingers one by one [Biswal1995][Friston1996]. The simplicity of the task is supposed to minimise the number of brain regions engaged in processing and executing it, allowing researchers to identify a singular brain regions purpose. A usual session consists of a number of sets of such a task, broken up by a second condition. That condition might be another task, or a resting phase, that is intended to contrast the patterns in brain activity provoked by the premier task and evade expectation bias [Kimberg2000]. Recurring patterns from premier set to premier set then suggest high task correlation. Experiment conditions like this are widely applied in research of the functional Connectome (or Functional Connectivity).

Lastly and conversely, Resting-State fMRI makes use of the same measuring techniques only without the subject performing a task. Subjects are instead asked to let their minds wander - activating the so-called Default Mode Network [Buckner2008]. It is proven, that even in a resting state the brains neural connections are still active [Greicius2003], 'linking' particular brain regions and thus creating a network, that is generally distinct from networks formed during engagement in tasks - making a 'mind-wandering' section an attractive alternating step in Task-Related fMRI studies.

2.1.2 Connectomics

Connectomics describes the study and eventual creation of high resolution wiring diagrams of the brain. This wiring or interconnectedness is agreed upon to be the defining feature of the nervous system and its mapping is only made possible due to the advancements in non-invasive MRI imaging technology.

The term Connectomics is a play on the word Genomics - suggesting that the Connectome is the foundational blueprint of the psyche like the genome is the blueprint of the body. As with the genome, the relevance and urgency of creating a high resolution Connectome is widely debated. However there is no doubt that recent advances in Connectome research have already contributed to Neuroscience's understanding of the brains inner workings.

Mapping the anatomical white matter structures of the brain for example has helped researchers identify abnormalities in Dementia- or Alzheimers-stricken patients - discoveries that might eventually lead to earlier diagnoses and possible treatments. Comparing full, anatomical Connectomes of healthy patients with those affected by neurodegenerative disease could be the key to understanding the origin of such afflictions.

Beyond the anatomical map of the brain, largely limited to the white matter responsible for linking brain areas, lies the mapping of the functional structure. Arguably more important to the understanding of the brains function is the actual activity and not the physiological structure, that acts as a medium for it. Enter the second major field in Connectomics: Functional Connectivity.

2.1.3 Functional Connectivity

Cortical functional connectivity, as indicated by the concurrent spontaneous activity of spatially segregated brain regions, is being studied increasingly because it may determine the reaction of the brain to external stimuli and task requirements. It is reportedly altered in many neurological and psychiatric disorders, which hints at the possibility of functional connectivity diagnosis as well as treatment [vandeVen2004].

Contrary to Anatomical Connectivity, which describes the actual structure of connections between neurons, Functional Connectivity is an entirely statistical concept. It describes a correlation or covariance of neuronal activity between two spatially remote areas [Friston1994] across a given time frame and in response to certain task stimuli or in their absence (see default mode network [Greicius2002]). As such it has no immediate reference to underlying structural connections. [Friston1996]

Areas of recorded neural activity are located in the grey matter in the outer layers of the brain and containing the neuronal cell bodies. Those areas range in size, depending on the experiment design and methods employed. With the ascent of high resolution imaging technology it is now possible to observe areas with voxel sizes of only a few cubic millimeters. However, the sheer density of neurons in these layers, with 3mm^3 -voxels of the cerebral cortex containing around 630.000 neurons (example calculation by the Geoffrey Aguirre Lab, 2012), still requires statistical information retrieval. Even at such a level of aggregation a study of the cerebral cortex will generate connectivity data for several tens of thousands of interconnected voxels. Connectivity data is usually obtained for all possible connections in a set of voxels, then thresholded and weighted with characteristics like persistence of the connection across time-series or condition instances as described in the fMRI section above. At the conclusion of such a study stands a weighted list of again, tens of thousands connections, a high weight suggesting a strong task-correlation of a distinct connection.

This data needs to be related to the actual spatial representation of the brain in order to determine major activation patterns and infer functional meaning. Of course simply linking every two-voxel connection with a straight line would result in a cluttered image void of any human-readable information. Luckily, the actual path is not a dimension of the information vector (also called *Connexel*, the basic unit of measurement in Connectomics [Worsley1998]), consisting only of two voxels and a weight of their connection - allowing for a large variety of visualisation methods, but lacking a standardised method.

2.2 State of Reasearch

2.2.1 Previous Work

The “pathless” nature of Functional Connectivity [Achard2006] is fertile ground for possible visualisation methods. The most important problem to solve is to visually or practically simplify the large volume of datapoints in a single set of measured connectivity without omitting essential information [Böttger2014]. While there is a definite standard for handling anatomical connectivity data (even with full datasets) [Pajevic1999], there is no method yet, that would visualise the entirety of a functional network. Commonly, visualisation of large scale functional networks is a two step process. First comes a preprocessing step of the actual input data, only second and much less well-developed is the visualisation step.
So far the most popular preprocessing approaches take into account only fractions of whole sets or subsample data from a larger set in order to “thin out” the connectivity data before eventually rendering it [Margulies2013].
There is a variety of methods, two/three of which will be described in the following paragraphs as they heavily inform the eventual visualisation process proposed in this thesis.

In the very popular *seed-based correlation analysis* for example, connectivity data of a single brain voxel, the seed region, is recorded by correlating every other voxel of the brains volume representation with the seed voxel across several time series [Biswal1995]. This approach is highly zoomed in and can offer a lot of detailed information, provided the researcher can make an informed a-priori decision on which voxel to designate as the seed voxel. The method is synonymous with the region-of-interest analysis (ROI) or cross-correlation-analysis (CCA), suggesting that the applicant will have to have decided on a region of interest before commencing their study. In most cases, researchers define a number of seed voxels in order to capture connectivity across major brain regions like the visual or motor cortex [Margulies2007][Greicius2003].

Independent component analysis on the other hand is performed on large swathes of data in order to identify major, mutually independent connective strands [Beckmann2005][DeLuca2006]. This allows the automatic identification of major, underlying network tendencies without a-priori knowledge of which regions they occur in. At the same time, this approach is highly zoomed out and runs the danger of dropping statistically minor but nonetheless significant connectivity information. As a statistical approach it is not suited to preprocess data for a full functional network visualisation as it filters out detailed elements of the connective strand.

Recently, different clustering algorithms have been used to identify major networks in raw connectivity data [Venkataraman2009][Lee2012]. Spectral clustering, as well as K-Means clustering algorithms have been employed and could successfully highlight well-known and therefore comparable structures of the default mode network, the visual cortex, the motor cortex and of the dorsal attention system. All of this without a priori knowledge of where these structures might originate [Venkataraman2009].

In most cases visualisation approaches for these highly preprocessed datasets are much less sophisticated. A general approach is the visualisation of functional connectivity in a simple connectivity matrix, marking connection strength, likelihood or prevalence between two given voxels by a colour hue **Fig.2.2** [Shirer2011] [Honey2007][Biswal1995].

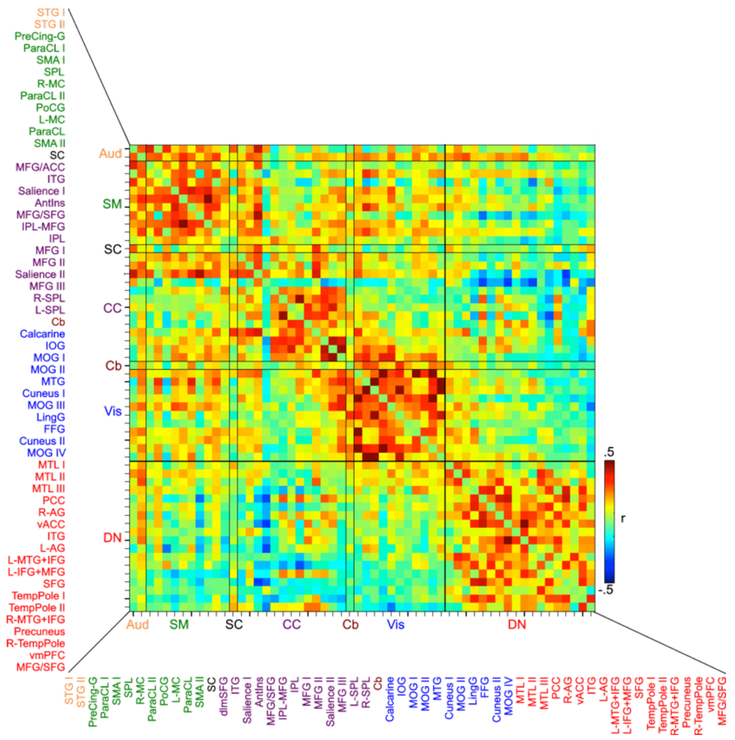


Fig.2.2: functional connectivity correlation matrix [Hutchison2015]. A red hue suggests high, task-related correlation. Both the x- and y-axes defined the same set of previously identified areas of interest.

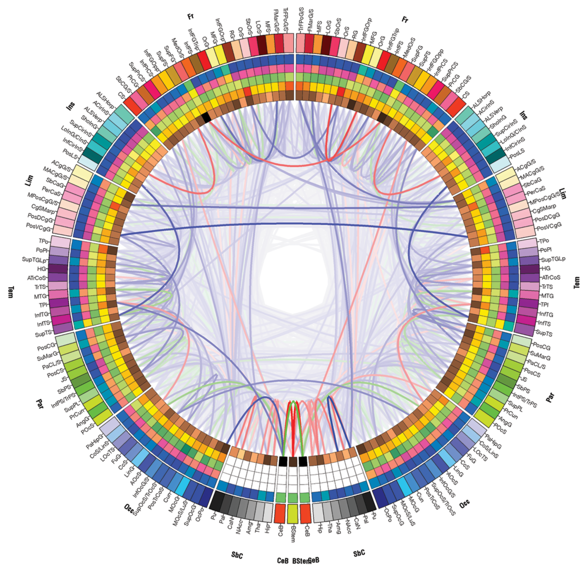


Fig.2.3: radial connectivity graph [vanHorn2012]. This graph connects areas of interest, that showed correlation during task-related fMRI. The alpha value of each edge is a measure of the connection strength.

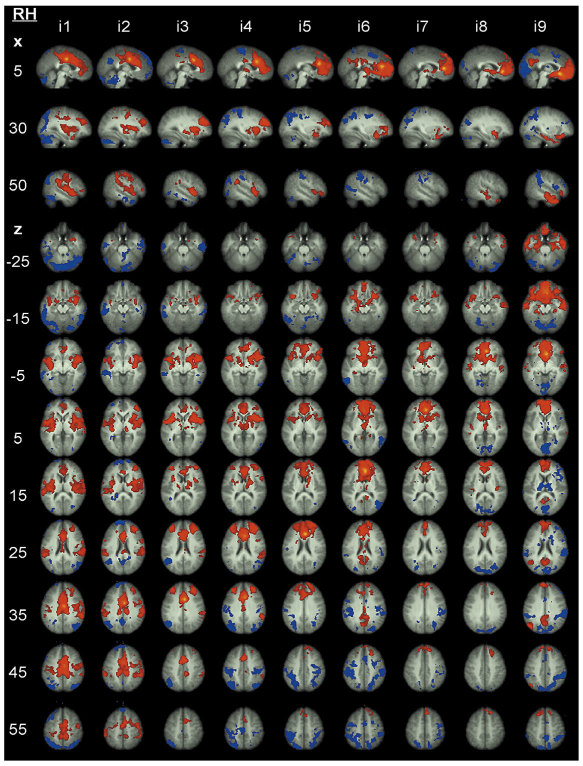


Fig.2.4: "Functional connectivity in right anterior cingulate cortex (inferior seeds)." [Margulies2007] The x-axis, labeled i1-i9 denotes a number of seed regions compared against a number of sagittal (vertical in relation to the head of the subject) slices on the y-axis.

Moving away from a strict matrix and towards graph-theory, several forms of diagrams have been proposed, for example two-dimensional connectivity maps [Achard2006]. Even with a disregard to the anatomical space however, graphs tend to get cluttered as the number of displayed connections increases. To resolve this issue, methods of visual summary such as hierarchical edge bundling [Holten2006] have been introduced to the visualisation process. Circular connection graphs for example, can provide a cleaner representation of network data employing such methods **Fig.2.3** [vanHorn2012][McGonligle2011]. Still, all of these approaches share a fundamental disadvantage: the inability to display connection data in relation to an immediate, anatomical reference because of their spatially abstract modalities.

Due to the relatively small size of connections aggregated in seed-based analysis approaches, visualisations do not tend to get cluttered and are largely created in two-dimensional space using fMRI generated image slices where connections are correlated with a colour value (usually blue to red, indicating probability of a recurring connection at this location) **Fig.2.4** [Greicius2003][Margulies2007][Biswal1995]. This visualisation method provides an anatomical reference in the form of actual brain scans of the subject. However it is not well suited to actually point out the connection between voxels, instead its main value lies in the information of how much a single voxel is connected to other voxels. Working with two-dimensional slices, identifying connections across depth levels is especially tricky.

2.2.2 Recent Developments

This section will summarise recent developments in preprocessing or gathering connectivity data and visualising it. These developments directly inform the approach of this thesis.

Another way of whole set preprocessing is the method of task-related edge density (TED), developed by Lohmann et al. and employed to gather the demo data for this thesis. In TED connectivity data is gathered on a per voxel basis, similar to a seed-based approach but for every given voxel in a volume representation. Every voxel represents a node and every possible connection an edge between two voxels. Connections with high correlation during task-performance are attributed with a high weight i.e. edge density. This approach does not require presegmentation of the data and allows the generation of high resolution functional connectivity network data, essentially identifying functional networks without omitting, summarising or singling out information. With this method, there is no prior filtering or simplification which leaves us with immensely large sets of relevant connections to visualise. Two methods of visualisation have been proposed in the original TED paper to deal with this issue.

First, a “hubness”-map, tracking only the number of connections for which a given voxel serves as an endpoint, this hubness value is encoded in the voxels colour. This of course does only represent the connectednes of a given region, not its actual relation to any number of spatially remote regions.

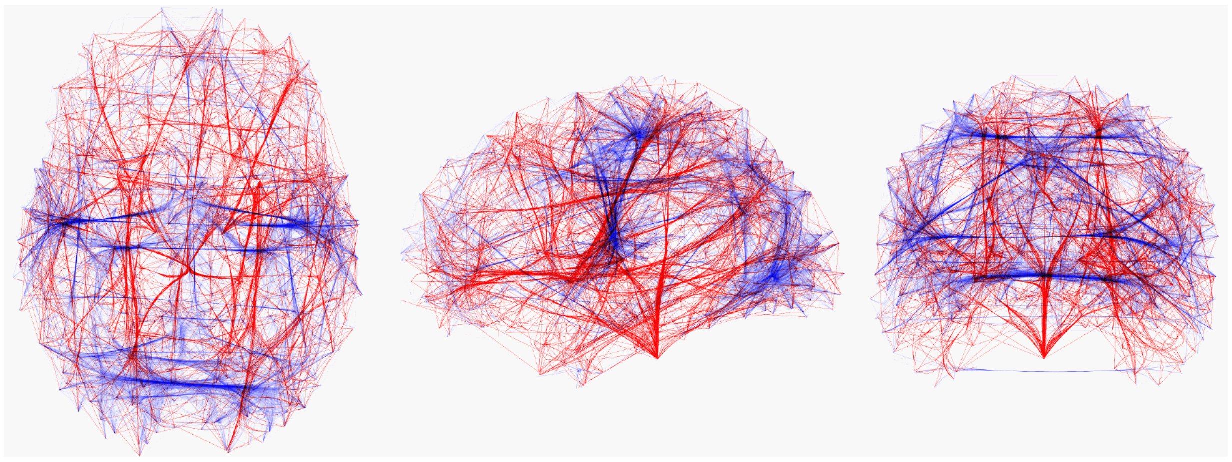


Fig.2.5, mean-shifted functional connectivity (blue) network visualisation, superimposed on visual approximation of structural network (red) [Böttger2014].

Second, displaying each connection as a line in 3d space using *braingl*. This does visualise an actual network of connective edges, but the high number of connections again obligates visual simplification. In this example, edges that are close to each other are bundled together, meaning their intermittent points are interpolated towards a shared central edge.

This bundling is based on a subdivision scheme called mean-shift, dividing the original data iteratively until visual inspection confirms a satisfactory division of all connections. Boettger et al. [Böttger2014] employ the mean-shift algorithm in order to determine high concentrations of connective strands and summarise them into logical and visual clusters. The big advantage of this method is, that the number of clusters does not have to be determined. Instead, the algorithm will approximate a set of convergent cluster centroids, or means. After the logical clustering is complete, each connective strand, starting out as a straight line, is interpolated with a close (closer than a self defined threshold value) mean edge in order to form bundles. This approach is successful in reducing screen-space cluttering and creates clear, visual network maps provided enough connective strands are drawn towards one of the specified means **Fig.2.5**.

It is also closely related to widely appreciated visualisations of DWI acquired anatomical connectivity data. Despite the success of this approach, it is still not possible to display the vast amounts of connections of an entire functional network without immense cluttering due to a number of issues. One being that a large amount of clusters would still result in many cross-sections of bundled connective strands, again rendering the visualisation barely readable. At the same time, reducing the number of clusters might result in overgeneralisation, bundling up connections that barely share attributes like end-point proximity or general direction. Boettger et al. conclude, that in order to produce a comprehensive visualisation of full scale connectivity networks one would have to follow "more radically different paths than nature" and incorporate methods like externalising the connective strands in a network map. This would also solve another issue. As argued by Margulies et al., the connective strands displayed in this approach are located inside the anatomical brain representation - possibly encouraging misinterpretations, that connective paths are part of the anatomic structure and not simply reflections of a high possibility of a functional connection.

2.3 Data Sources

2.3.1 Experiment

The experiment of which the demonstrative data is used in this thesis centres around a set of face recognition tasks in order to derive functional connectivity data with an explicit reference to memory access. Condition one requires participants to recognise a single, specific face in a set of faces shown one at a time. Condition two again provides a set of faces, only now participants must identify the second to last face they have been shown. Each condition occurs four times each, without a discernible pattern in order to minimise effects of expectation bias and maximise reformation of the functional networks.

fMRI data is gathered using a prototypical 9.1 tesla fMRI, which yields very high resolution segmented images of a brain volume. The acquired data is preprocessed using a standard pipeline for noise reduction and subsequently analysed using the task-related edge density method as described in the previous section.

The experiment is conducted by Dr. Johannes Stelzer, member of the High-Field MRI group at the Max-Planck Institute for Biological Cybernetics. At the point of the conclusion of this thesis, the results of the study have not been published.

2.3.2 Data Types and Domains

The goal of the experiment is to aggregate task-state functional connectivity data for large-scale connectivity, i.e. connectivity on a whole-brain basis. A single dataset consists of multiple files, the most relevant being a 3D matrix containing the actual connection data.

1. Every datapoint in the connection data matrix contains three values. The first and second value are indices referring to voxel coordinates in the brain volume, the third value is a floating point number denoting the connection weight between the two voxels. These datapoints, which hold the information on a single connection of correlating or covariant regions have been termed "connexel" [Worsley1998], as in pair of 3D-positions. This data is the result of the TED analysis and is delivered in .csv format.

2. The connection data stores connections as pairs of indices, each referring to a voxel coordinate inside the brain volume representation. The brain volume is represented by a 4D matrix essentially storing said indices behind three nested index values. These values in turn refer to a three-dimensional voxel coordinate. This data source can be understood as a voluminous box, separated into voxels of size 1,2mm^3. Only the coordinates, that are part of the brain volume inside the box hold an index value and as such can be easily extracted. Volume data is held in .nii format, commonly used in neuroimaging **Fig.2.6** Left.
3. Third is the visual brain representation in form of a simple 3D mesh of the scanned brain. In its role to serve as visual/anatomical reference for connectivity data it is an essential part of the visualisation - especially in this approach of visualising functional connectivity data in relation to anatomical space. The whole-brain mesh representation is in .vtk format **Fig.2.6** Right.

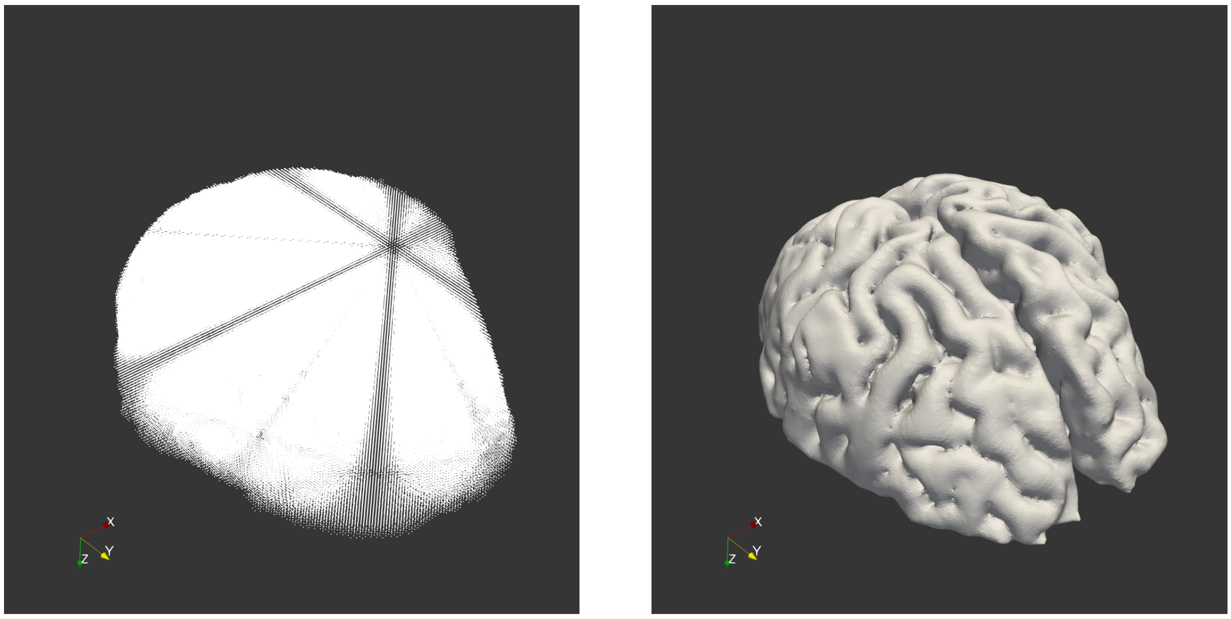


Fig.2.6: Left volume representation. Right .vtk mesh representation. Note, the volume representation contains a dense matrix of points. Because of the matrix's symmetry, the impression of a superimposed pattern appears.

These three files are closely related and should be used discretely for each subject, or single study. Since functional connectivity is derived from grey matter activity in the outer layers of the brain it is common, that datasets are prepared with an “inflated” representation of the subjects brain. This essentially reduces the depth of the Sulci (the burrows between the folds of the brain surface) allowing for a less ambiguous view at grey matter activity. Especially when applying a clustering algorithm based on the distance of 3D coordinates, depressed Sulci can lead to a perceived proximity of connection endpoints.

Working with an inflated brain representation will reduce logical errors when calculating distances between connection endpoints and as such is paramount in order to make the application of a distance based clustering algorithm valid. The same is true for the application of a preprocessing method, involving distance as a relevant attribute.

After preprocessing there is not much relation between actual connectivity data and the volume representation, since we expect, that connectivity information has been gathered correctly and with respect to a possibly inflated state. However, using an inflated volume representation in conjunction with a non-inflated whole-brain mesh will lead to possibly harmful and certainly false misinterpretations of the visualisation results.

2.4 Analysis of Requirements

2.4.1 Identification of User Groups

Based on the background information provided above, the following chapter will outline a set of requirements that need to be fulfilled in order to create a viable visualisation toolset.

The set of functional requirements is three-fold. As a user software, the visualisation tool prototyped in the course of this thesis is to be applied by people other than the original developer. Furthermore it is to be open-sourced, allowing third parties to continue the development process at any point. Lastly, the visualisation itself is meant to be interpreted by the original operator as well as peers, interested in the field of connectivity research.

All things considered, it is possible to identify three distinct user groups, two of which are addressed in the immediate scope of this thesis:

Operators of the software. Experts and researchers in the field of neuroscience or medicine and knowledgable in the general application of advanced and professional user software in those fields. Abilities in computer science and programming should be beneficial but not required.
Peers in the field of connectivity research. This includes students, researchers and others interested in the topic of functional connectivity. This user group is not concerned with operating the software, but instead with interpreting the results. An expected scenario involving this user base would be the distribution of a research paper featuring visualisations generated with the provided software.

Excluded from the immediate scope of this thesis is the following and last group.

Third-party developers. Generally well versed in Python programming and the use of common packages. Basic understanding of matrix and vector calculation is necessary, as well as an all-around good read on algorithms. Not more than a passing interest in neuroscience is required.

2.4.2 Functional Requirements

As the user group analysis suggests, the following section will be divided into requirements of operators of the software and requirements of “consumers” of the softwares visualisation output. The second set of requirements can be understood as a combination of factual validity and value of the visualisation.

Application Requirements:

- The paraview filter should have a basic user interface. K-means operates on the basis of a constant k number of clusters. Operators need to have the option to manipulate k from inside the user interface in the case of live calculations. Additional attributes to manipulate the visualisation could be the subselection of edges, a switch to display or not display edges and cluster selection.
- In addition to the previous requirement, users should be able to explore hierarchically generated K-Means data by scrolling along the hierarchy with another ui-element.
- Inputting the data into the user interface (by file path) is the only requirement to generate a visualisation.
- Calculations of clusters should be efficient enough to be processed live and in no more than a few minutes, even for medium size datasets. Small datasets should be processed in mere seconds. This feature is intended to serve as a scouting process of where and how to visualise connectivity data - large scale datasets should be processed in the command line interface.
- The software has to offer the option of loading precalculated cluster data from file in order to reduce system load. To meet this requirement a simple command line interface is required, such that users of the software can generate data before using the paraview integrated filter.

Visualisation Requirements:

- Central element of the visualisation is the display of the mean edge, marking an average connection edge for a cluster of connections and located on the outside of the anatomical brain representation.
- Clusters have to be visually distinct from one another, this will be achieved with a continuum along three attributes: colour, projection radius and interpolation along edge points towards the respective mean edge. This is the essential step of applying the information retrieved from the K-Means processing results.
- One of the defining factors of the underlying data in this visualisation approach is its high resolution. All of this data should be represented in the final visualisation output, allowing the observer to explore the full spectrum of a whole-brain connectivity graph.
- The visualisation has to be accurate in its representation of connectivity endpoints. Every visualised edge has two endpoints, originally provided by the data input and marking the actual correlating/covariant regions in the brain. Through all preparatory and visualisation processing steps this data has to be left unchanged and displayed as provided.
- There has to be clear reference to the anatomical background of the connection data. This could be a transparent whole-brain scan, a point cloud or a background image.

2.4.3 Technical Requirements

Similar to the manyfold nature of the functional requirements, the technical requirements could be separated into two categories. Namely, a set of requirements for the clustering step and another for the actual visualisation step.

Technical Requirements for Clustering:

- The nature of a single connective edge requires a special distance function inside the K-Means algorithm. The algorithm should therefore be simple enough to expose a distance function and make it interchangeable.
- Connective edges are six-dimensional, consisting of two three-dimensional points. In order to compute the distance between two edges accurately the distance function has to be agnostic to the direction of each edge.
- The final software bundle created in this thesis should provide a simple Unix executable command line script, that will allow operators to run K-Means and hierarchical K-Means on any set of two-point edges and save the output to file.
- The K-Means algorithm will be implemented in a naive fashion and should therefore not require Python packages beyond Numpy.

Technical Requirements of the Visualisation

- As with the original distance calculation, the six-dimensional properties of each edge could lead to a faulty visualisation. Edges have to be aligned before they can be displayed in order to avoid visual loops and crossing lines.
- Interpolation between original connective edges and mean edge has to accept several different interpolation methods in order to provide control over level and progression of interpolation in the bundling step.
- To create intermittent points for every two-point connective edge the Rodrigues rotation must be implemented.
- The radius of the edge projection for each cluster needs to be calculated based on the average distance between this clusters endpoints.
- To make hierarchical data explorable, the original edges that make up a cluster on a current hierarchy level must be retrieved. To retrieve edges an algorithm has to follow along the hierarchy.
- All of the above must be implemented with VTK and its Python bindings.

2.4.4 Demarcations and Limitations

- All calculations (especially the K-Means algorithm) will require Numpy as a dependency for matrix calculations and essential, mathematical operations.
- The only efficiency factor addressed in visualisation processing will be that the task can be completed in linear time without overburdening the systems RAM. Large datasets should be precomputed in a system Python environment and not inside the Paraview client, i.e. the programmable filter.

2.5 Environment

2.5.1 Paraview and VTK

VTK (Visualization Toolkit) is an open-source software package for 3D computer graphics, image processing and visualisation. VTKs main component is a C++ Library, which is made accessible to other programming languages through *interpreted interface layers.* The software prototyped in this thesis will employ VTKs Python interface layer, which will allow easy access to code sources and a faster prototyping environment. Additionally, developing in Python will grant access to a variety of useful libraries like Numpy.

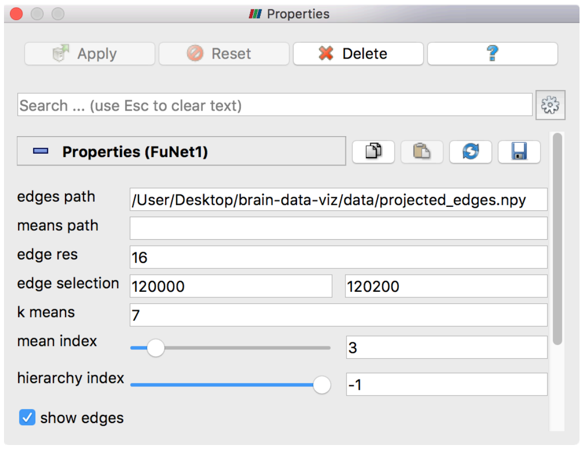
Paraview is a software tool for data analysis and visualisation. Built on the 3D-rendering pipeline and general functionality of VTK, Paraview effectively acts as a user interface for VTK and enables users to interact with its tools through graphical interface elements.

Users can load 3D-models into the application and start manipulating edges, vertices and faces. Unlike regular 3D-modeling software, Paraview does not allow the direct manipulation of either of these 3D-elements through cursor but instead offers a range of filters, which perform operations on them algorithmically. Paraview offers several versions of filters, some of which generate data while others require a data input of some sort. Some filters can also be applied on plain data, like a simple data table, allowing the generation of 3D meshes or transforms from regular numerics.

VTK offers several classes for data handling. Most of these classes consist of a number of cells and some added attributes and methods. Cells represent sets of vertices, edges or faces and are necessary to display 3D objects. VTK makes a distinction between topological and geometric data. Cells provide the topological element but hold no immediate information on their position in 3D space. Instead, every cell refers to one or more elements in the 3D object's 'points' array, which holds a set of plain, geometric coordinates. A vertex refers to a single point in the array and its topology has zero dimensions, an edge refers to at least two points and therefore has upwards of one dimension and so on.

Because of the open-source nature of both VTK and Paraview the development of custom plugins (in most cases filters) is encouraged by extensive documentation and a variety of possibilities. The most accessible option to create a custom filter for example, would be to use the "programmable filter". It can be applied to any VTK data source (mesh, plain data and others), exposing said datasource to a Python environment in which it can be manipulated before it is returned to the rendering pipeline. This allows users to directly run Python code using the VTK-Python library inside Paraview and on a specific data source.

However accessible and immediate, a programmable filter cannot be understood as distributed software, since it would require users to copy and paste the actual Python source into the filters “script” property. In order to expand a simple Python source into an easily integrable plugin an XML file, meeting Paraview standards, is required. This XML file is interpreted by Paraview as a filter or data source and can therefore be loaded using the integrated plugin manager. It can also hold a layout for the properties panel of the filter it represents, offering a way to include manipulable fields or other GUI-elements in a plugin while hiding others in an "advanced" section **Fig. 2.7**.



edges path /User/Desktop/brain-data-viz/data/projected_edges.npy
means path
edge res 16
edge selection 120000 120200
k means 7
mean index 3
hierarchy index -1
☑ show edges

Fig.2.7: programmable filter attributes exposed through XML. Paraview allows a set of simple input types, like fields, range sliders and checkboxes.

The prototyped package will provide the Python source of a programmable filter as well as a readymade XML filter plugin, using a similar Python source as its hidden script property. Additionally, the XML filter will be exposing access-required attributes as described in section 2.4.2 in its graphical user interface. This will allow operators to control the visualisation properties on the fly and without touching source code. The exact layout of these files will be discussed in the main section.

2.5.2 Systems and Software

All of the Python libraries listed below are effective dependencies, meaning that the full spectrum of the provided software can only be used if all of them have been installed and properly sourced in the users Python environment. This holds true for the application of the XML filter inside Paraview.

Software:

- Python 2.7.10, universal high level programming language, developer and proprietor: Python Software Foundation, https://pythonprogramming.net
- NumPy, Python library for multidimensional array computing and general math appliances, developer and proprietor: Travis Oliphant, Numpy Team, http://www.numpy.org
- Matplotlib, Python Library for the display of function graphs, developer and proprietor: Michael Droettboom, http://matplotlib.org
- NiBabel, Python library for nifti/.nii and other neuroimaging file reading and writing, developer and proprietor: https://www.nipy.org
- VTK, Visualisation Toolkit C++ Library for 3D-Rendering with Python-bindings, developer and proprietor: Kitware Inc., http://www.vtk.org
- Paraview 5.4.0, Open-Source Application for data analysis and 3D visualisation built on VTK, developer and proprietor: Kitware Inc., Sandia National Laboratory, Los Alamos National Laboratory, https://www.paraview.org

Hardware:

- Desktop PC with Intel Xeon CPU W3690 @ 3.47GHz x 6, 23,5 GB RAM, Ubuntu 14.04
- MacBook Pro 2015 with Intel Core i5 @ 2,7GHz x 2, 8GB RAM, macOS Sierra 10.12.6

3. Edge Visualisation

3.1 Problem Statement

There is a large number of possibilities of drawing lines in 3D space. Due to the immense number of connections in a single dataset of functional connectivity however, the proposed visualisation method was confined to a set of requirements.

Previous work has demonstrated, that straight lines, connecting the endpoints of connexels create a densely cluttered visualisation [BrainNetViewer] [VisualConnectome] - even at much smaller datasets. Readability of a high resolution dataset, like the benchmark dataset in this thesis, would in no way be possible without extensive downsampling of the data or other undesired methods of preprocessing. The same goes for bundled, but still, internally displayed connective lines, as proposed by Boettger et al. [Böttger2014], since the number of clusters would likely still be enough to clutter up the brain volume and obfuscate experiment data. Therefore the connective lines drawn with the implemented algorithm are external, surrounding the brain mesh at a near constant distance to the surface. This approach is comparable to a visualisation proposed by Foucher et al. [Foucher2005], which so far has not been employed on whole-brain functional connectivity data.

The second requirement is scalability. This visualisation is supposed to display high-field functional connectivity data with hundreds of thousands of connective edges. A single line drawing pass should be efficient enough to allow the exploration of the data without excessive processing times. Additionally, simplifying the line drawing algorithm will improve readability and allow the continuation on the development of the project by developers to come.

Lastly, one of the major achievements of high-field functional connectivity is its voxel resolution, meaning that active brain regions can be singled out at 1,2mm^3 steps. Being one of the defining factors of such datasets, it is paramount to preserve the possibility of localising these connection endpoints in the final visualisation.

The following section will examine how these requirements are intended to be fulfilled with the proposed, multi-step visualisation model. A simple method of drawing lines at constant radius in 3D space will be proposed as a foundation for the visualisation. These connective edges will eventually link the two endpoints of each connexel in the dataset. Lines drawn in this way will lie on a sphere, surrounding the brain mesh. In order to approximate the set of lines more closely to the brain mesh surface, a projection method will be introduced in the second subsection. This method will be capable of translating any number of coordinates between a spherical and an ellipsoidal representation.

3.2 Drawing Connexel Edges in 3D Space

With a prepared set of pairs of connection endpoints, everything is properly set up to draw edges between those points in 3D space. In order to draw an edge, Paraview expects a number of 3D coordinates to draw line segments between. The idea is to generate a set of 3D coordinates approximating a curve by a set of segments connecting the two endpoints of each connexel at a constant radius, thus generating a connective strand for each relevant connection. The number of intermittent segments determines the apparent smoothness or resolution of each edge. To draw the lines at a constant radius, two approaches were tested.

The first attempt was to convert both endpoint coordinates v_o, v_d (origin and destination) of a connexel from a cartesian coordinate representation to a spherical one, then generate a number of points n along the curve by linear interpolation between the two points' ϕ and θ angles, while leaving the radius r untouched. Projecting both origin and destination coordinates would enable the definition of a constant radius, allowing a very simple interpolation scheme, with $V(v_o, v_d, n)$ being a set of n 3D coordinates between the two connexel endpoints

$$v_o, v_d(x, y, z) \to v_o^s, v_d^s(\theta, \phi, r)$$
$$V(v_o, v_d, n) = \{v_i : v_o + (v_d - v_o)\left(\frac{i}{n}\right) \forall i \in \{1, \ \dots, n\}\}$$

This approach however presented itself as problematic, because of the domain of the polar angle θ inside the spherical coordinate system. θ has a domain of $[0;\pi]$, so intermittent points tended to interpolate predominantly on the azimuthal angle ϕ when θ angles of the endpoints were particularly close. This resulted in an unexpected path of the curve on the sphere **Fig.3.1**.

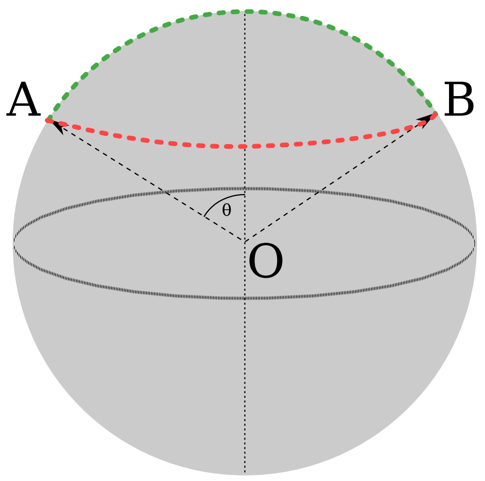


Fig.3.1: Interpolated edges in in polar coordinates, approach one, linear interpolation between angles in spherical coordinate space (red) with the undesired interpolation. Intuitively, the shortest path (green) would be expected.

The interpolation of polar coordinates does not link the two points on the sphere surface $A$and B, using the shortest path. One way to ensure that the shortest path is always taken is to remain on the plane defined by AOB. This can be achieved using the Rodrigues Rotation formula. This formula rotates a vector v around an axis k (given by the normal of the plane AOB) by an angle θ.

$$v_r = v cos\theta + (k \times v) sin\theta + k(k \cdot v)(1 - cos\theta)$$

where v_r is the vector, rotated around the axis k by the angle θ.
This solves the previous issue and yields 3D edges as expected. A few additional methods were required to successfully implement the Rodrigues formula. First, k had to be defined, around which the origin vector, the first of the two endpoints of the connexel, could be rotated. This could be achieved using the cross product of the two endpoints (origin and destination), thereby finding a perpendicular unit vector k to the plane in which both vectors are included.

$$k(\overrightarrow{AO}, \overrightarrow{BO}) = \begin{cases} \dfrac{\overrightarrow{AO} \times \overrightarrow{v}_{rand}}{\|\overrightarrow{AO} \times \overrightarrow{v}_{rand}\|}, & \text{if } \overrightarrow{AO} \times \overrightarrow{BO} = \overrightarrow{0} \\ \dfrac{\overrightarrow{AO} \times \overrightarrow{BO}}{\|\overrightarrow{AO} \times \overrightarrow{BO}\|}, & \text{else} \end{cases}$$

In the case of the origin and the destination vector being collinear, taking the cross product of either one and any random vector, that is not a multiple of either, would return a perpendicular vector. This auxiliary method has been implemented recursively to ensure the generation of a non-collinear cross vector.

The second important auxiliary method aims to find the angle θ between the two points' unit vector representations, again assuming the plane AOB.

$$\theta = \arccos\left(\frac{v_o}{\|v_o\|} \cdot \frac{v_d}{\|v_d\|}\right) = \arccos\left(\frac{\overrightarrow{\mathrm{AO}}}{\|\overrightarrow{\mathrm{AO}}\|} \cdot \frac{\overrightarrow{\mathrm{BO}}}{\|\overrightarrow{\mathrm{BO}}\|}\right)$$

This angle delta can be divided into any number of chunks, granting control over the edge resolution once again **Fig.3.2**.

Any number of connexels can easily be represented by an edge in 3D space employing the methods above. Projecting edges with constant radius will essentially be create on a sphere, exclusively holding all edge vertices. In order to connect each edge to its two endpoints inside the brain volume it is sufficient to simply add them to the array holding the intermittent vertices. The result is accurate, in that it connects two regions of activation visually without obscuring their original location. However any edge holding coordinates at constant radius as well as the two original voxel coordinates will have a pronounced right angle, due to the outward projection **Fig.**

3.3.

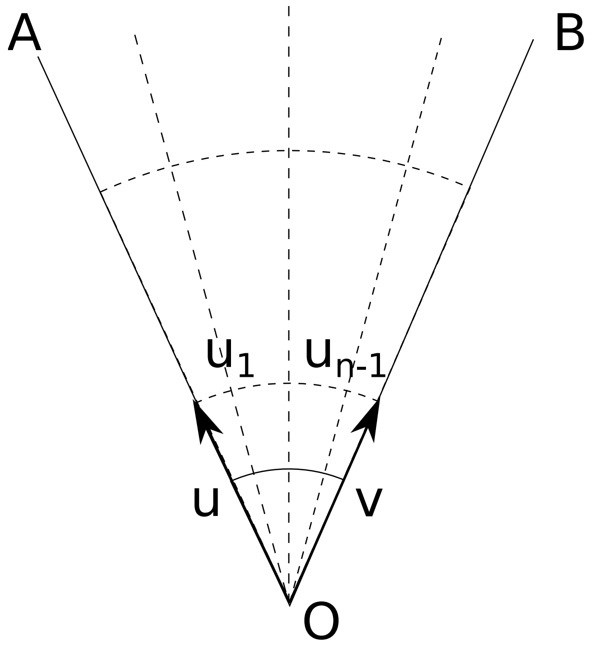


Fig.3.2: Rodrigues Rotation a unit vector u towards a unit vector v. Intermittent vectors {u_1, ..., u_{n-1}} can be projected to any radius by simple multiplication.

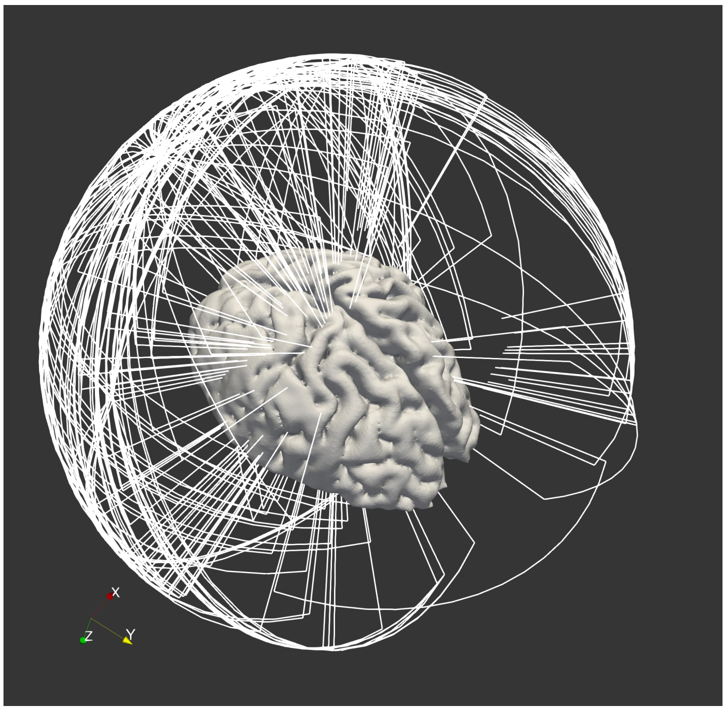


Fig.3.3: connective edges projected at constant radius. The stilt-like line segments connecting the original endpoints to the projected vertices are identifiable by the right angles.

3.3 Spherical Representation and Matrix Projection

The previous section discusses in detail how 3D edges at constant radius are created for each connexel in a dataset of functional connectivity. The goal however is, to draw those edges at a close-to-constant distance from the brain mesh surface in order to relate their orientation to possible pathways more closely.

During the line drawing step all edges were generated on a sphere, the shape of the brain mesh however is closer to an ellipsoid. While still an approximation to the mesh surface, projecting the connective edges from a spherical representation to an ellipsoidal one, resembling an upscaled version of the brain mesh, will allow for a better approximation of a constant distance to the brain surface. The process of projecting points into a spherical representation will be termed "forward-transform" or "forward-projection" and the process of projecting from a spherical representation to an ellipsoidal one "back-transform" or "back-projection".
The forward transformation can be easily achieved through shrinking along the ellipsoid's axes. The back transformation will be achieved by multiplying along the same axes by a factor inverse to the forward transform. Information on both the axes and the factors can be determined by computing the covariance matrix of the coordinates forming the ellipsoid in question. In this case the covariance matrix will be computed using the vertex coordinates of the brain mesh.

Let p_i be a single vertex coordinate defined as

$$\mathrm{p}_i = (\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i)$$

The covariance will be computed using either combination of the coordinate sets x, y and z, herein represented by placeholders a, b. The covariance matrix Σ_P of the entire set of centred vertex coordinates $\bar{\mathbf{M}}^p$ will then include the covariance of all possible combinations of coordinate sets, with regards to the sequence.

$$\mathrm{cov}(\mathrm{a}, \mathrm{b}) = \frac{\sum_{i=0}^{n} (\mathrm{a_i} - \bar{\mathrm{a}})(\mathrm{b_i} - \bar{\mathrm{b}})}{n - 1}$$

$$\Sigma_P = \begin{bmatrix} \mathrm{cov}(\mathrm{x},\mathrm{x}) & \mathrm{cov}(\mathrm{x},\mathrm{y}) & \mathrm{cov}(\mathrm{x},\mathrm{z}) \\ \mathrm{cov}(\mathrm{y},\mathrm{x}) & \mathrm{cov}(\mathrm{y},\mathrm{y}) & \mathrm{cov}(\mathrm{y},\mathrm{z}) \\ \mathrm{cov}(\mathrm{z},\mathrm{x}) & \mathrm{cov}(\mathrm{z},\mathrm{y}) & \mathrm{cov}(\mathrm{z},\mathrm{z}) \end{bmatrix}$$

The forward-projection can be applied to both, mesh vertex coordinates and connexel endpoint coordinates held in the volume representation. This transformation has the advantage of making the data isotropic, such that no particular axis/direction is pronounced. The volume information was transformed, prior to clustering due to the expected improvement of clustering results. Instead, this adds falsehoods to the results, because during the clustering step the data is abstracted from its actual, physical dimensions obscuring distances and therefore cluster layouts. This should be avoided in further application of the software.

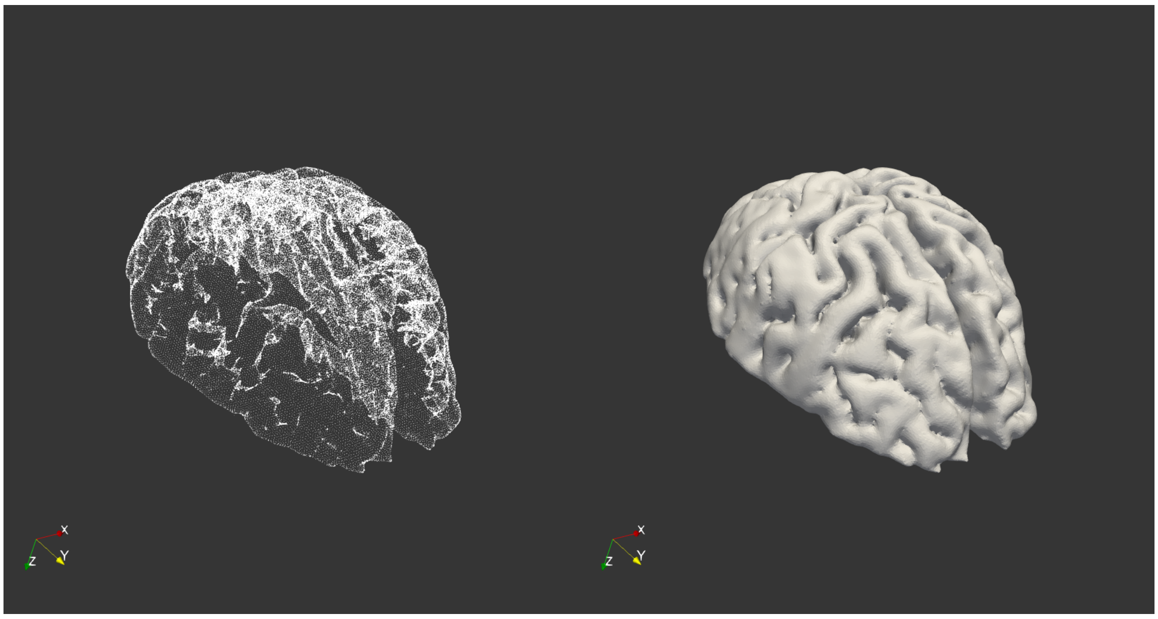


Fig.3.4: side by side of volume information represented by a point cloud and brain mesh surface. The volume information needs to be understood as a tightly-knit three dimensional matrix of points. It holds about six times as many vertices as the mesh surface.

The matrix Σ_p is real positive definite and hence can be diagonalised, and written in the following way:

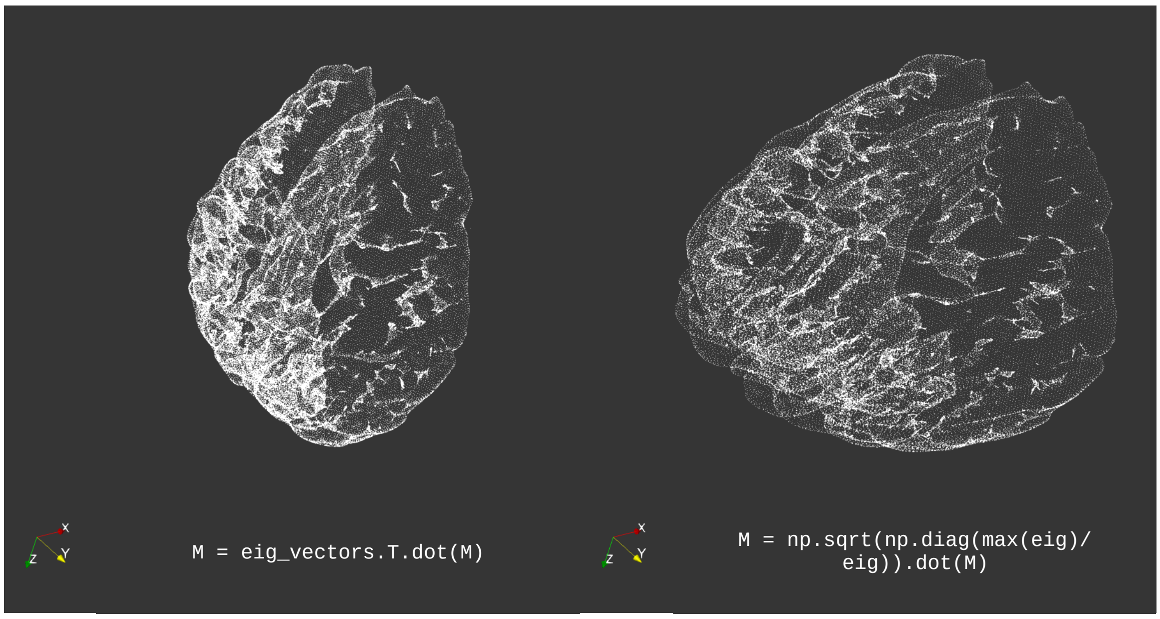
$$\Sigma_p = \mathrm{R} \cdot \mathrm{D} \cdot \mathrm{R}^t$$

where matrix R is a rotation matrix and the matrix $\mathrm{D} = \mathrm{diag}(\lambda)$ is diagonal. R contains the eigenvectors of Σ_p. The eigenvectors define the axes of the spread of points in the brain mesh, and the set of eigenvalues λ the associated variance of the spread. The eigenvector with the highest eigenvalue therefore defines the axis with the most spread. The intention is, to now rotate the centred volume information $\bar{\mathbf{P}}$ of the brain to align it with the spread of the mesh points, using the transpose of the 3×3 matrix of eigenvectors R^t. The result will be the same set of points, projected onto a new coordinate system basis $\bar{\mathbf{P}}^r$ **Fig.3.5** left.

$$\bar{\mathbf{P}}^r = \mathrm{R}^t \cdot \bar{\mathbf{P}}$$

The coordinates now need to be scaled by yet another matrix. For scaling we use $\mathrm{D}^{-\frac{1}{2}}$ multiplied by a constant factor $\max(\lambda)$, such that the scale of the original brain mesh is preserved **Fig.3.5** <u>right</u>. After execution of this forward transformation, the connective edges can be drawn for any set of connexels as seen in section 3.2. such that,

$$p_i^s = \frac{1}{\max(\lambda_i)} \mathrm{D}^{-\frac{1}{2}} \mathrm{R}^t p_i$$



```
M = eig_vectors.T.dot(M)
```

```
M = np.sqrt(np.diag(max(eig)/
       eig)).dot(M)
```

Fig.3.5: <u>left</u>: volume information rotated by transposed matrix of eigenvectors, <u>right</u>: previously rotated volume information scaled by inverted and normalised diagonal matrix of eigenvalues.

The volume data represented in this way, all intermittent edge coordinates can be created and added to the transformed set of the original coordinates.
With all vertices, including the intermittent edge coordinates, represented in the same space, a back projection of the entire set is possible in one go. This can be understood as the inverse of the process described above and the same covariance matrix of the brain mesh is required.

$$p_i = \mathrm{R}\mathrm{D}^{\frac{1}{2}} \max(\lambda_i) p_i^s$$

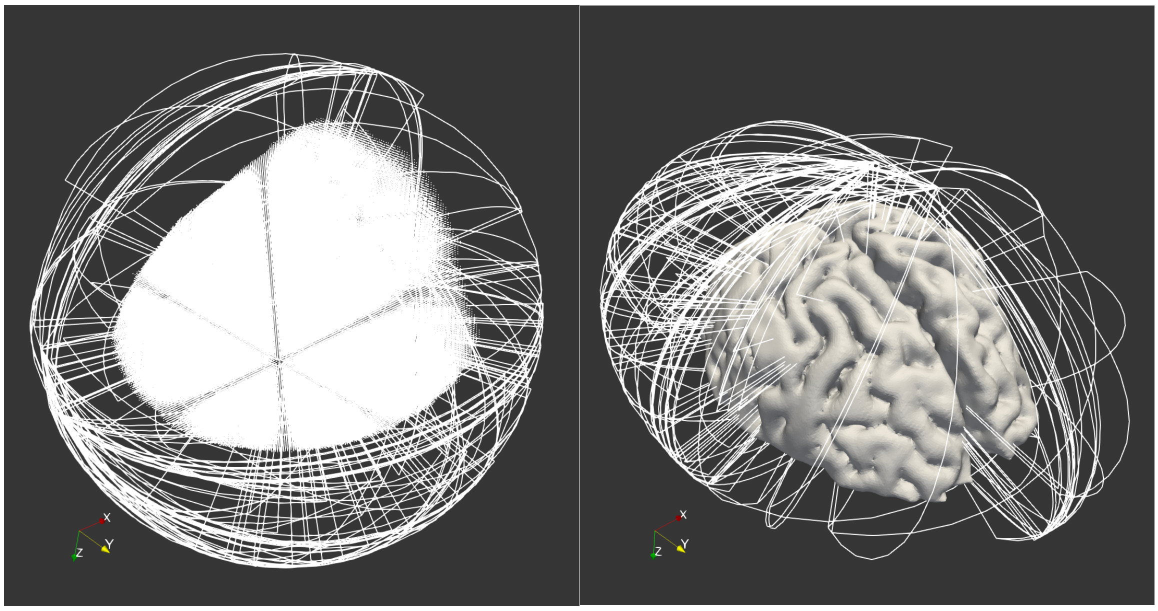


Fig.3.6: progress of drawing and projecting generated lines as well as volume information. Even at a small subset of n=128 connections (<0.2% of the whole dataset), the visual output is densely cluttered.

The point coordinates in $\bar{\mathbf{P}}$ have been projected into spherical space, resulting in a spherical representation of the entire set $\bar{\mathbf{P}}^s$. Let $\dot{\mathbf{P}}^s$ be a copy of $\bar{\mathbf{P}}^s$ with the newly generated edge coordinates. To project them into the ellipsoidal representation of the brain mesh they need to first be scaled. To achieve this, the inverse of the diagonal matrix originally used to scale the set of connexel coordinates is required. In this case, this is rather unintuitive, because the matrix would be the inverse of an inverse diagonal matrix of eigenvalues. The result will be a version of $\dot{\mathbf{P}}^s$, rotated to the original rotation.

A set of random edges has been created in a spherical representation of the brain volume and consequently been projected to match the brain surface in the following example **Fig.3.6**.

The Python implementation for both the forward projection matrix and the back projection matrix uses Numpy to compute the eigenvalues of a covariance matrix, passed as an arguement. It is therefore:

```python
def forward_projection_matrix(cov_mat):
    eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)
    eigen_matrix = np.sqrt(np.diag(max(eigen_values) * eigen_values**-1))

    m = eigen_vectors.dot(eigen_matrix).T
    return m


def origin_projection_matrix(cov_mat):
    eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)
    eigen_matrix = np.sqrt(np.diag(max(eigen_values) * eigen_values**-1))

    m = np.linalg.inv(eigen_matrix).dot(eigen_vectors.T).T
    return m
```

Note, that this implementation does not use the exact reciprocal of the eigenvalues in its `eigen_matrix` variable. This is to maintain the general scale of the point matrix as seen in **Fig.3.5**. Taking the reciprocal instead would significantly down-size the entire matrix. While not problematic after back projection and to the visualisation in general, this might impair debugging during development.

3.4 Conclusion

In conclusion, using the above methods, edges can be projected into an ellipsoidal representation, such that they have a close to constant distance to the brain surface. The method is efficient as it is a simple matrix multiplication, that can be performed with accelerated routines (Blas, Numpy). Performing this method on large numbers of edges at once is not an issue.

Combined with the method of edge drawing at constant radius, explained in the beginning of this chapter, the pipeline for drawing lines is kept computationally minimal, while still being visually accurate.

Of course, an ellipsoid is only a very rough approximation of an actual brain's surface, since it does not take the highly folded structure of the brain into account. The folds of the brain also add significant variance to the brain mesh coordinates, which might impact the ellipsoidal approximation.

While many edges can be processed in this way, only the next section will introduce a method of displaying them without extensively cluttering the screen-space.

4. Clustering Connective Curves

The previous section focuses on the display of connective edges, however it does not address the issue of scalability. Drawing a set of only 128 edges can result in a cluttered visualisation and the entire dataset holds more than a thousand times that many edges. This section will introduce a method of clustering, that will be the foundation of the visual summary of the data. Clustering is a method of identifying a set of objects, that share a higher similarity amongst each other, than to objects of a different set. In this case, edge clusters will be defined by the relative closeness of their endpoints.

4.1 K-Means Algorithm

K-Means is a clustering method, that aims to partition a set of n d-dimensional datapoints into k clusters, such that the in-cluster variance is minimised. In the analysis of 2D- or 3D-points for example this would mean, that K-Means seeks to deploy its cluster centroids (elements, that share the dimensionality of the dataset, also termed 'mean') in such a way, that aggregated, squared distance between datapoints and their assigned cluster centroid is at a minimum.

General steps of the K-Means algorithm are as follows:

1. Initially, the k cluster centroids are provided to the algorithm. A common approach is to sample these randomly from the dataset, that K-Means is performed on, however there are more elaborate methods of initialisation which seek to improve the overall computation time and quality of the clustering.

The following two steps are repeated until some condition occurs. In most implementations this is the convergence of the cluster centroids, to a point where there is no more or insignificant change between iterations.

2. Each datapoint in the set is assigned to its 'closest' (attribute-wise) cluster centroid.
   Given a set of k cluster centroids $c_i, \dots, c_k$ the assignment step would be

   $$S_i^{(t)} = \{p : \text{dist}(p - c_i^{(t)}) \leq \text{dist}(p - c_j^{(t)})\ \forall j \in \{0, \dots, k\}\}$$

   where $S_i^{(t)}$ is a set of datapoints associated with cluster $c_i^{(t)}$ at iteration t and p is a single datapoint. dist() is a distance function. So each cluster is a set of elements, that share a higher likeness to their centroid, than any other cluster centroid.

3. Each cluster is composed of a subset of the original data. To conclude a single step in the iteration, each clusters centroid is updated with the aggregated values of its particular subset.

$$c_i^{(t+1)} = \text{barycentre}(S_i^{(t)})$$

where for every cluster centroid $c_i^{(t)}$ the updated version $c_i^{(t+1)}$ will be the centre of mass of the subset $S_i^{(t)}$.

4.2 Adapting K-Means to Connexel data

K-Means generally uses the euclidean distance between the datapoints it is performed on.
Each connexel datapoint can be encoded in a six-dimensional vector containing two 3D-coordinates. The arising problem is, that the order of the coordinates becomes a factor in the eventual result of the cluster analysis when it should not. An example in two-dimensional space will illustrate the problem:

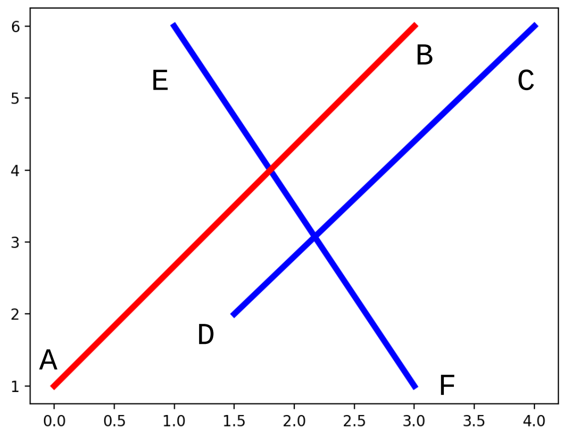


Fig.4.1: faulty clustering of a set of edges. n=3, k=2

The two major functions of K-Means, namely the distance function and the barycentre function, have to be modified to accommodate the nature of the present dataset.

4.2.1 Distance Function

The example makes it immediately obvious, that edges AB and CD should belong to the same cluster. Due to the application of a euclidian distance function, coordinates A and C as well as B and D have been compared resulting in much higher squared distance between endpoints than expected.

This highlights, that the distance of two edges is defined by the distance *d* between their two respective endpoints, so there are always two possible comparisons for any given edge in euclidian space.

$$d = \| \{x_1, y_1, z_1, x_2, y_2, z_2\} - c_i^{(t)} \| \quad or \quad \| \{x_2, y_2, z_2, x_1, y_1, z_1\} - c_i^{(t)} \|$$

where the set $\{x_1, \ldots, z_2\}$ is a single six-dimensional datapoint consisting of two 3D coordinates, compared against a centroid $c_i^{(t)}$ of the same dimensionality. So for every datapoint the distance to each centroid has to be computed for a flipped version p^f as well.

$$S_i^{(t)} = \{p : (\|p - c_i^{(t)}\|^2 \wedge \| p^f - c_i^{(t)}\|^2) \leq (\|p - c_j^{(t)}\|^2 \wedge \| p^f - c_j^{(t)}\|^2) \; \forall j \in \{0, \ldots, k\}\}$$

One of two heuristics can now be selected for the assignment step, either compare all minimum distances between edges and centroids to determine the set affiliated with a specific cluster or compare all maximum distances. While the former one is more intuitive, it is important to note, that the choice of which distances to compare does not have an effect on the outcome; the important part is the consistency across all comparisons. This effectively makes the assignment step of the algorithm agnostic to the orientation of its datapoints, resulting in the expected outcome.

4.2.2 Barycentre Function

The orientation of the lines would also have an effect on the update step of the cluster centroids, if not explicitly handled otherwise. It is therefore necessary to preserve the information of which version (flipped or original) of a specific edge the mean centroid has been compared against, when the edge was assigned to its cluster. The edges will have to be aligned accordingly, before recomputing the centroid **Fig. 4.2**.

The desired computation for the cluster centroid of $C_0 = \{AB, CD\}$ should therefore look like this

$$c_0 = \left(\frac{A+D}{2}, \frac{B+C}{2} \right)$$

as opposed to

$$c_0 = \left(\frac{A+C}{2}, \frac{B+D}{2} \right)$$

This is achieved by storing a flip-map in the form of a boolean array during distance computations. Before cluster centroids are updated, this array is used to switch the first three elements in a datapoint with the last three, or leave it unchanged as required.

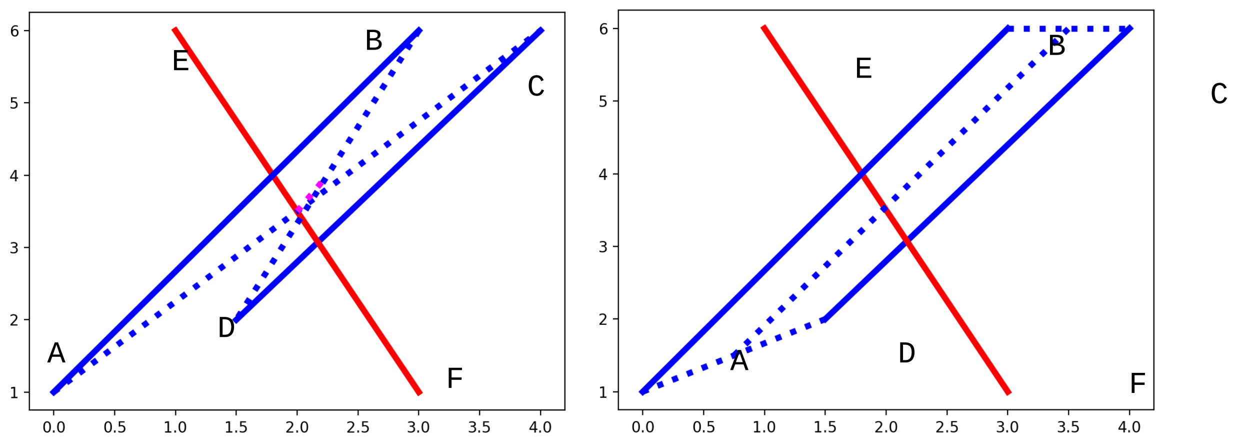


Fig.4.2: barycentre calculation. Left faulty recomputation without prior alignment of edges. Right expected computation of cluster centroid, averaging

4.2.3 Results

Results indicate the success of the methods described above. The within-cluster sum of squared distances is decreased significantly when compared to the approach using euclidian distance computation. In one hundred sample computations of $k = 16$ and $n = 1024$, for a shuffled dataset, the average sum of squared distances was ~27998mm for the adapted and ~34668mm for the euclidian approach. This divergence grows as k gets larger.

Generally the adapted method reduces the amount of outliers in any given cluster, essentially creating more representative cluster centroids **Fig.4.3**.

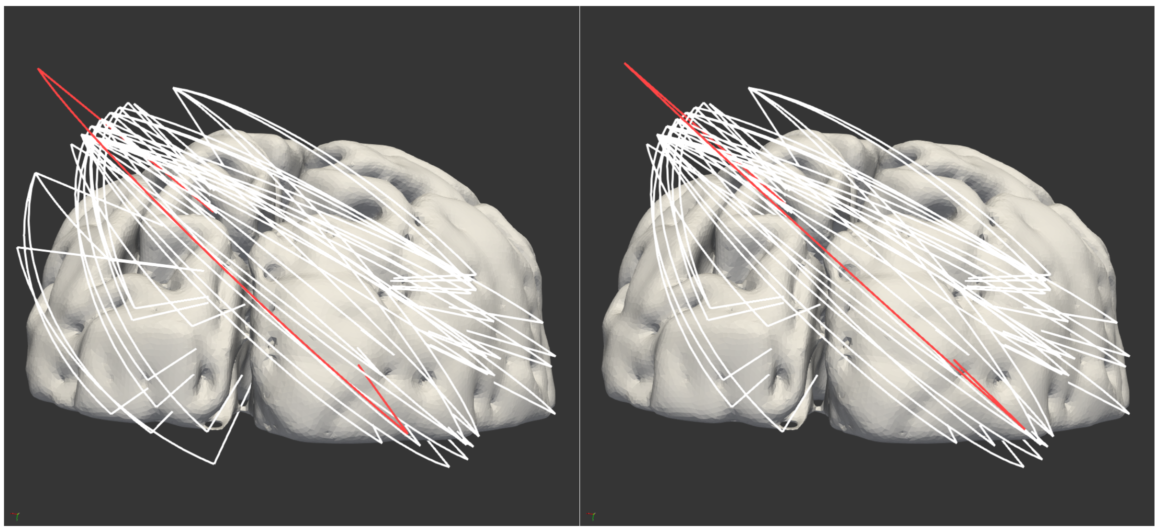


Fig.4.3: cluster centroid (superimposed in red) of a single cluster. Left the basic, euclidean distance function includes a set of outliers (mid left). Right direction-independent, adaptive approach assigns outliers to different clusters. The centroid is a more accurate representation of the clusters centre of mass.

4.3 Hierarchical Scheme

K-Means is successful in summarising large datasets in the form of a predefined number of clusters. The requirement for the a-priori definition of a number of clusters however is one of the algorithm's major limitations. Especially when processing large datasets with significant processing times, converging on a representative number of clusters can be time-consuming and inefficient.

A way to make the exploration of even large datasets a more interactive experience is, to apply K-Means hierarchically. A hierarchical scheme will consist of several applications of K-Means, where only the first considers the original dataset of connexels. Subsequent applications will be performed on the converged centroid edges of the previous pass, while also reducing the number of clusters *k*. Each pass is stored and can be traced back along the hierarchy to the original dataset **Fig.4.4**.

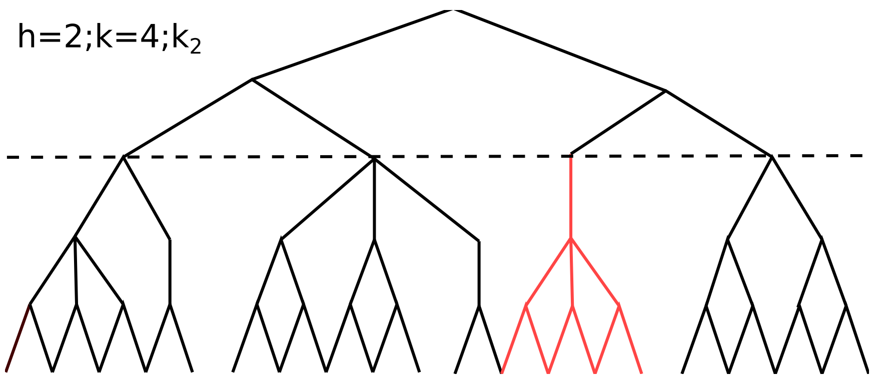


Fig.4.4: example of a hierarchy and cluster selection. Assuming hierarchy level 2 and cluster two are selected, the edges on the bottom of the highlighted path will be displayed as a single cluster.

Not only does this enable the exploration of many different K-Means results at different numbers of clusters, but it also bears potential to identify clusters more consistently. Especially when examining a dataset one cluster at a time and moving up and down the hierarchy, there is a high probability of identifying a highly effective cluster layout, meaning number and distribution of clusters, for a particular subsection of the data.

The design of this hierarchical application scheme is simple. It iterates entire K-Means applications using the proposed implementation, with the number of clusters given by

$$k^{(t+1)} = \frac{n}{x^{t+1}}$$

where t is the current iteration step in the hierarchy, n the number of edges in the original set and x a user defined value marking the divisor in between iterations. For $x = 2$ for example, the benchmark set of 186266 connections would take 16 iterations to reach $k = 2$, allowing for many hierarchy levels to be created and inspected.

As mentioned, only the first pass considers the original dataset, so every subsequent set is defined by the set of centroids produced in the previous pass;

$$\mathrm{A}^{(t+1)} = \{c : \mathrm{barycentre}(\mathrm{S}_i^{(t)})\}$$

where A^{t+1} is a set of edges given by a set of centroids with each centroid c given as described above (reference formula step 2 from K-Means).

These centroids are a summation of the clusters they represent, which depending on the hierarchy level, consist only of previous centroids. The eventual goal is to relate every hierarchy levels cluster layout to the original dataset, therefore every cluster needs to be followed along the hierarchy, to aggregate all connective edges represented by the leaves of the hierarchy graph **Fig.4.4**.

Without taking all edges into account during each pass this method runs the danger of steering centroids away from actual cluster centres of the original data. So when relating upper hierarchy levels to the original dataset, outliers and higher aggregated squared distances are to be expected. Additionally, since the hierarchical process forces K-Means to be executed on progressively smaller datasets and lower numbers of clusters, relatively independent clusters may be summarised in a single cluster of a higher order.

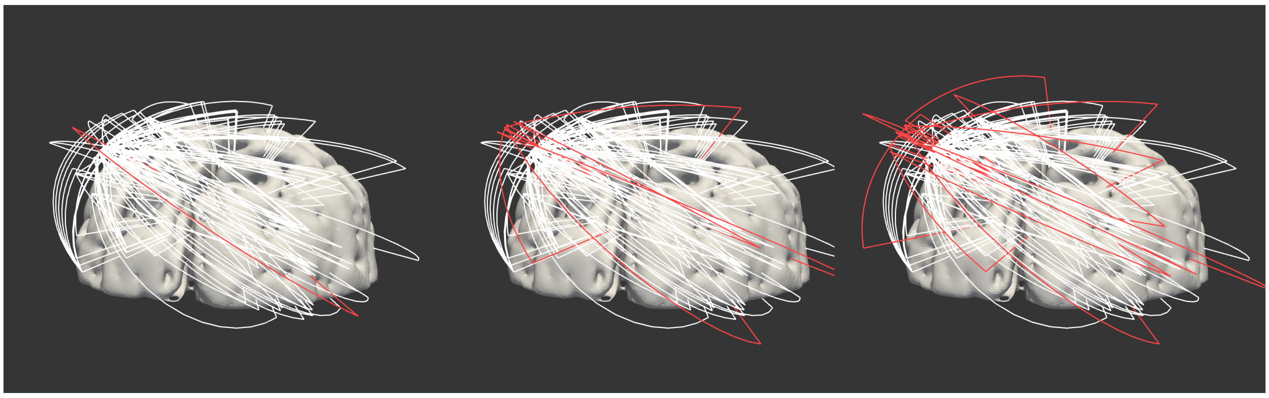


Fig.4.5: three hierarchy levels dividing the same cluster. The hierarchical clustering scheme has been applied to the same dataset as above. The cluster has been chosen, due to its similarity to the cluster in the last section. From left to right, cluster hierarchy moves from toplevel (6) to bottom level (0)

Fig.4.5 shows a single cluster in three different hierarchical stages. Comparing this outcome with the one shown in the previous section, it is easy to tell that this cluster contains a larger number of outliers. Indeed, the aggregated squared distance in the example of $k = 16$ and $n = 1024$ is across a number of 100 trials about 1.3 times as high as in the single pass clustering.
This rate is relatively consistent between different size datasets and different numbers of clusters and numbers of hierarchical passes. This concludes, that hierarchical K-Means does not achieve better clustering layouts on higher hierarchical levels. However, with access to the hierarchy attribute inside the user interface, operators can skim through a dataset at different cluster layouts with next to no delay.

4.4 Colours

The clustering results will not only enable methods of edge bundling, but will allow the colouring of the connexel edges for improved distinction between clusters. In the field of functional connectivity research, colours are generally used to encode further information of the datasets. Some prominent features like connection strength, dominant orientation of the connection or general connectedness are common colour scales. The only predictor of colours in this example will be the index of each edges associated cluster.

In the application of the hierarchical scheme, the integration of colour as a cluster property is especially effective in analysing network tendencies **Fig.6**.

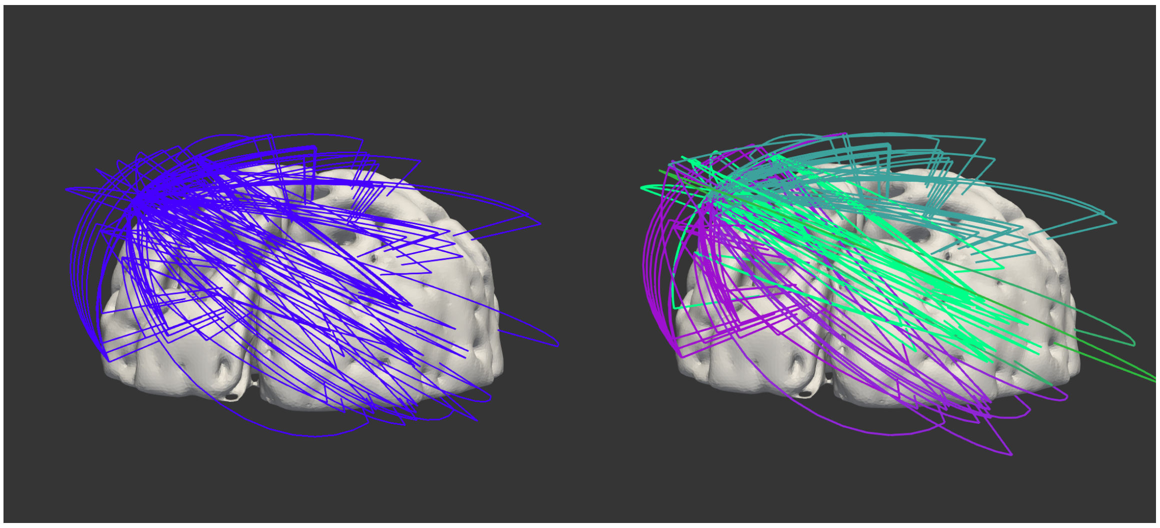


Fig.4.6: colour as a cluster property at different hierarchical levels. The K-Means results used in this visualisation are equal to the ones in the previous example. Left cluster at top hierarchy level, single colour. Right cluster at hierarchy level 3, the same level as the mid example in Fig.x

4.5 Performance

K-Means as well as Hierarchical K-Means have been implemented with little regard to efficiency. The major goal was to create an easily readable implementation, that could compute a result for a dataset of any size in linear time.

In some cases this lead to the inability of using more advanced methods of computation as provided by Numpy. Numpy would allow the computation of squared distances between an entire set of edges and an entire set of centroids at once, so the first approach was the following:

```
---
distances = ((edges - centroids)**2).sum(axis=1)
---
```

However, the large number of connective edges in the benchmark set would lead to an immense size of the `distances` matrix.
With $k = 16$ for example, there would of course be 16 centroids, so 186266*16 distances, times two because of the computation of both the original and the flipped version. A single distance is stored as a 64bit floating point number. For a single step of K-Means, this computation would occupy upwards of 23 Gigabytes of RAM. A single pass distance calculation was therefore out of the question. Instead distances were then calculated per edge, slowing down general performance but allowing for the computation of a set of any size on machines with much less RAM. This implementation will allow a Single K-Means computation of the entire set to be completed in linear time.

Random initialisation of centroids leads to relatively high variance when it comes to processing times and number of K-Means steps required in generating a result. The variance is reduced relative to the number k of clusters. Generally, using a higher number of clusters will increase the duration of every K-Means pass but decrease the number of passes required. Despite this, the main predictor of processing times is the size of the dataset n.

Other more methods of initialisation would likely reduce the variance of results and more efficient implementations could reduce processing times by a large margin.

4.6 Conclusion

The K-Means algorithm has been successfully implemented and customised to meet the requirements of a six-dimensional dataset in the form of functional connectivity data. While there were no exact metrics for the accuracy of the clustering, except for squared distance aggregation between approaches, visual confirmation suggests, that clusters are computed as expected - with a high priority for similarly oriented connexel edges.

Both K-Means and hierarchical K-Means have been identified as valuable preprocessing methods. While the single pass variant delivers results with more efficient cluster layouts, the hierarchical scheme contributes in that it provides explorable datasets, which can inform decisions about relevant data subsets or clustering layouts for future computations.

While the results can be made explorable in Paraview's user interface, they do not provide realtime visualisations and require, depending on the size of the source data, several minutes of preprocessing.

The results are nevertheless foundational in their importance for the eventual visualisation of whole-scale connectivity data. Computing a single pass K-Means and displaying only the resulting centroids is already a method of reducing the density of a high field dataset with a strong approximation of a lossless summary. Another method of exposing major network tendencies is the introduction of colour to the edge visualisation. Basing each edges colour on the index of the associated cluster can allow a large variety and enhance the visualisation in many ways.

The main purpose of the clustering however, is the visual summary of all connexel curves and not just a subset. The following section will examine bundling methods, to create a concise and informative visualisation.

5. Edge Bundling

The cluster analysis described in the previous section is the foundation of the visual summary of the dataset. In functional connectivity research datasets can reach sizes of upwards of 100.000 connections for a whole-scale functional network. The experimental dataset, used as a benchmark in this thesis has an even higher resolution. Where previous attempts at visualising functional connectivity have generally taken only subsets into account, this visualisation aims to prototype a method of displaying a full dataset, paying special attention to the preservation of valuable details like the origin and destination locations of each connexel. The proposed method of visual summary is a bundling method, which 'ties' edges, associated with a specific cluster, together in order to de-clutter the visualisation and incorporate hundreds of thousands of connexels in a single view.

5.1 Bundling Heuristics

After the generation of edges in 3D space and the computation of K-Means, the results can be incorporated into the visualisation in order to be able to display high resolution functional connectivity data.
The bundling is achieved by interpolating each connective edge towards its associated cluster centroid. Again, the prime directive is to leave the endpoints of each edge untouched, such as to not manipulate the actual experiment data. Additionally this will ensure, that despite visual summary around the centre of each edge, the high resolution of the origin and destination neighbourhood is preserved and visualised accurately.

After back projection to brain mesh space, the following can be applied for every edge.

$$\mathrm{e}_c = \begin{cases} [0,1] \to \mathbb{R}^3 \\ t \mapsto \alpha \mathrm{e}(t) + (1-\alpha)\mathrm{c}(t) \end{cases}$$

where e_c describes an edge, affiliated with cluster c_j in the form of a set of 3D coordinates. Every edge is interpolated vertex-wise towards c, the cluster's centroid edge, by a coefficient in the open unit interval contained in the set α. All sets introduced hold the same number of elements. Making α a function of time will allow the implementation of different, more elaborate interpolation functions.

5.2 Interpolation Functions

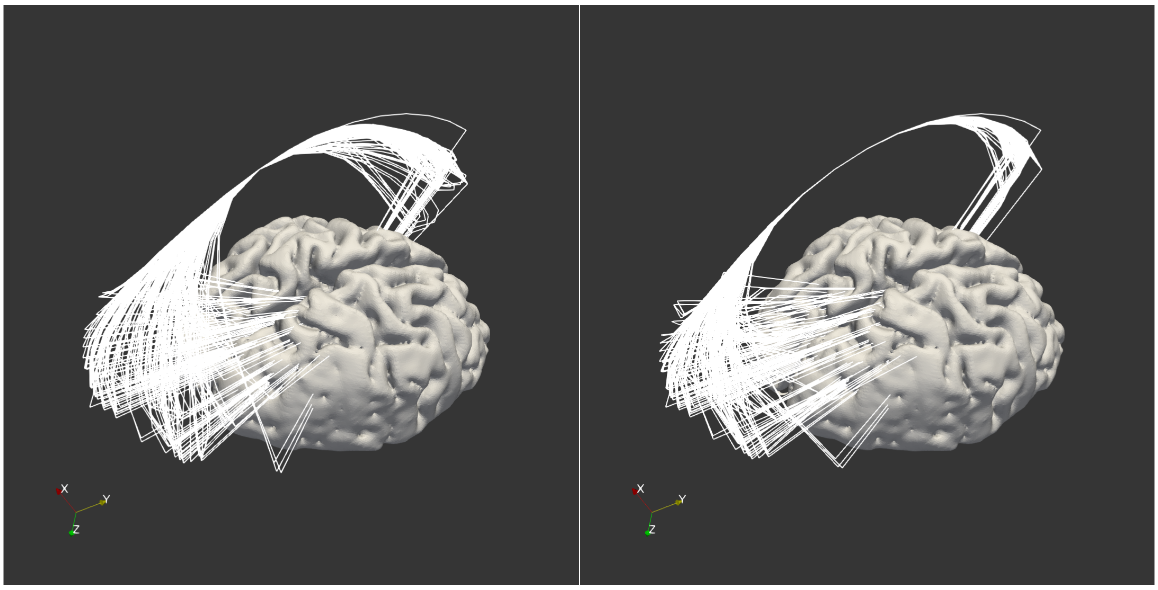


Fig.5.1: examples of bundling applied to a single cluster. Function graphs are given below.

α can be initialised with numerous functions, so long as they satisfy the following properties:

$$\alpha : \begin{cases} \alpha(0) = \alpha(n) = 0 \\ \alpha(t) \in [0,1] \end{cases}$$

α has to retain the constraint, that $\alpha(0) = \alpha(n) = 0$, such that the interpolation of both connexel endpoints equals zero; leaving them entirely unchanged. This is necessary to not
manipulate the original dataset. Another requirement of α is its symmetry. In order to have a symmetric interpolation, the zenith of α should be at its centre with each value before or after being lower.

Depending on current visualisation requirements clusters can be converged quickly or in a more gradual manner using a variety of functions **Fig.5.1**.

The function in the visualisation on the <u>left</u> for example lets edges converge slowly and is given by

$$\alpha = \{x : sin(x \cdot \frac{\pi}{|\mathbf{s}|})\}$$

where $\mathbf{s}$ is given by $\mathbf{s} = \{0, \ \ldots, n-1\}$ and n is a number corresponding to the total number of vertices held in any given edge/centroid.

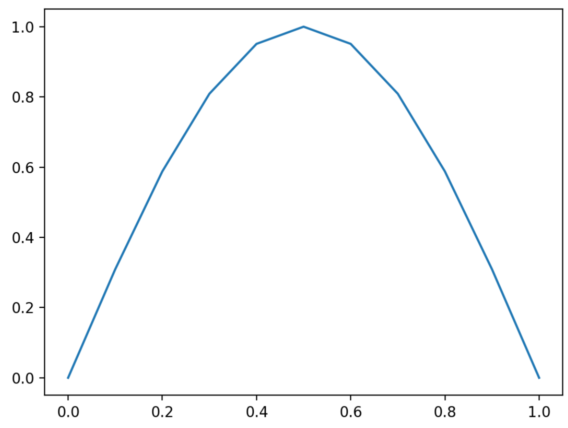


Fig.5.2: gradually converging alpha function for the unit example of n=1. This function corresponds to the visualisation on the left.

The function on the right converges more quickly and is given by

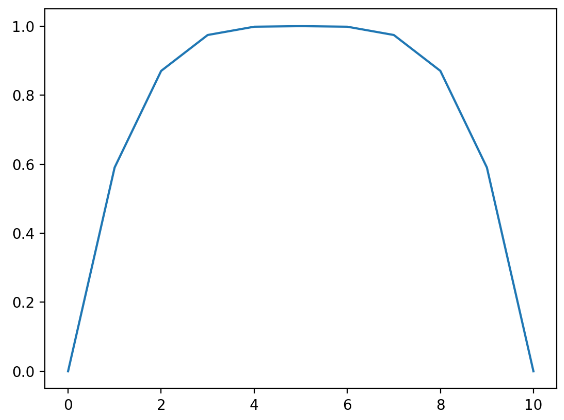


Fig.5.3: quickly converging alpha function for a concrete example of n = 10. This function corresponds to the visualisation on the right.

$$\alpha = \{x : 1 - \frac{(x - \frac{n-1}{2})^4}{n^{\log(-\frac{n-1}{2})^3}}\}$$

In addition to the steepness of the interpolation function, simply multiplying the resulting set of coefficients by a value in the interval $(0{,}1) \in \mathbb{R}^+$ will offer some control over the level of compression around the centre of the bundle. Any value lower than 1.0 will prevent a full convergence of connective edges and centroid edge, allowing a loser bundle visualisation that preserves information on origin and destination of a single connection. This information will be forfeited when the bundle converges on a specific point around its centre.

5.3 Bundling Refinement

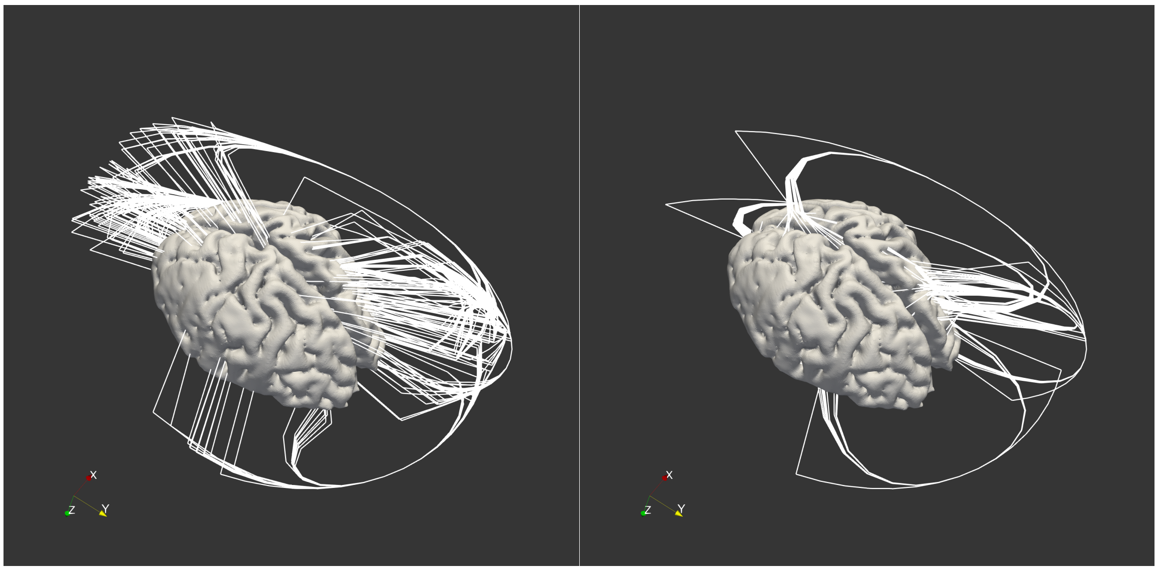


Fig.5.4: two bundling archetypes. Left: Simple interpolation of connective edges towards a mean edge with a similar radius . Right: interpolation of connective edges with a radius towards a mean edge with significantly higher radius .

Simply interpolating edges in their current state is a fast method of visually summarising the connectivity data. While not without merits, improving on this method will be necessary to attempt to satisfy the requirements laid out in the introduction of the visualisation, namely creating a visualisation without clutter, which allows the display of any number of connexel curves without obscuring each connexels endpoints.

Interpolating edges towards centroids makes cluster orientation and spread of endpoints obvious, but exact endpoint locations can be obscured by the 'stilts' connecting the endpoints to the first points of the connective edge laying on the ellipsoid **Fig.5.4** left. Despite these drawbacks this type of visualisation could hold some value. For example when approximating an ideal number of clusters, as variance among endpoint neighbourhoods is easily visible in this approach.

A way to reduce the 'stilt' effect, is to drastically minimise the projected radius of the edges, while keeping a high radius on the centroids. This will reduce the magnitude of the curves 'stilt', effectively turning it into a much less significant segment of the curve. Post-interpolation, the edge will now approximate a more continuous curve **Fig.5.4** right. Especially in combination with a quickly converging interpolation function, this reduces the screen space each cluster inhibits significantly.

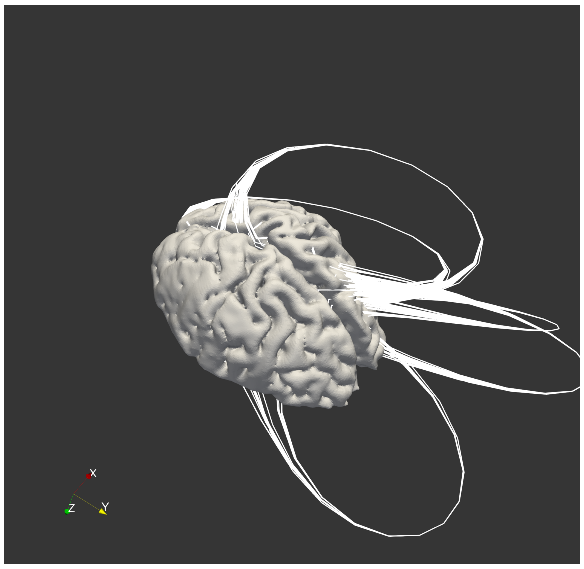


Fig.5.5: continuous radius. Improved reduction of spread at endpoint neighbourhoods, mean edges now blend in with connective edges.

A more solid and intuitive solution is the implementation of a function, that gradually increases, then decreases the radius of a given line during its course. This function delivers an array of radii, reaching their zenith at the central coordinates of any edge - similar to the interpolation set α introduced above. The resulting edges behave more smoothly and also reduce line spread as the edge bundle approaches either endpoint neighbourhood **Fig.5.5**.

This would turn the radius into a function of time, such that

$$\mathrm{e}_r = \mathrm{r}(t)\mathrm{e}_c(t)$$

where e_r denotes a single edge, whose vertices have been projected using a continuous radius function $\mathrm{r}(t)$. This function interpolates the radius between the inherent radii of the connexel endpoints v_o, v_d over a user-defined zenith or maximum radius. The continuous radius function is therefore given by the following

$$\mathrm{r} : \begin{cases} \mathrm{r}(0) = r_o, \mathrm{r}(n) = r_d \\ t \mapsto (r_o + (r_d - r_o)(\frac{t}{n-1})) + z(t) \end{cases}$$

where r_o, r_d are the radii at the origin and destination point of a connexel curve, n is the number of vertices and $z(t)$ could be given by any version of $\alpha(t)z$ or a similar function.

The general goal is to externalise connexel curves, so a radius, larger than the maximum extent of the brain mesh on any axis should be chosen for the user-defined zenith. The maximum extent of the brain would be half its length, with the length being the axes of the highest spread of vertex coordinates. Because of the original 1:1

scale of the dataset, in this example the zenith should be defined as at least 75mm. The visualisation in **Fig.5.5** uses a zenith of 120mm.

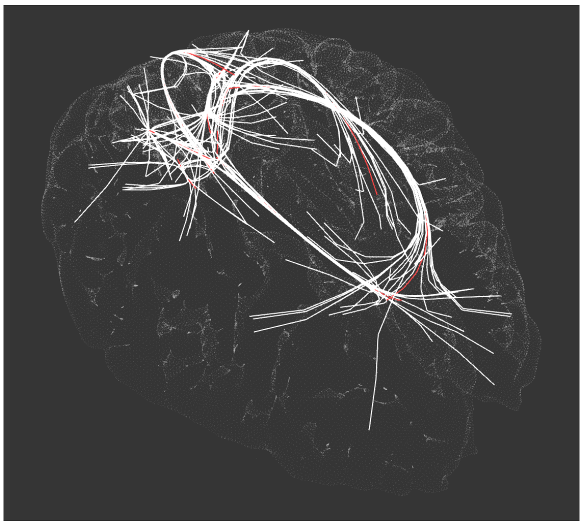


Fig.5.6: internalised connective curves. Effect of continuous radius in combination with a low zenith.

An interesting side-effect of the continuous radius function is, that choosing a small zenith of only 1mm will still result in an accurate visualisation, preserving the endpoint coordinates. The bundled edges are displayed on the insight of the brain, similar to the visualisation proposed by Boettger et al. [Böttger2014] **Fig.5.6**.

5.4 Conclusion

It has been established, that even simple interpolation of edges, projected at close-to-constant radius will achieve visual clutter reduction. This method furthermore indicates the spread of endpoints at a clusters neighbourhood, suggesting that, while not quite desirable for whole-scale visualisations, it might have value in identifying a more adequate number of clusters for a given set.

The method has been improved upon in two steps, first, edge vertices have been created at a low radius, while the centroid edges, towards which they were interpolated, kept a relatively high radius. This reduced the angular effect of the curve and turned a perceived 'stilt' into a less significant element of the entire curve. Because of the requirement of a specific relation between connexel and centroid edge radii another method was prototyped.
Instead of two constant radii to interpolate between a continuous radius function was developed, that smoothed out the curve approximation. This approach was even more successful at reducing clutter around endpoint neighbourhoods and improved the interpolation.

6. Software Package

6.1 Package Contents

The final software package contains a number of scripts written in Python as well as a testing suite for the provided functionalities. This will ensure stability in future developments and extensions of the package.

The following paragraph will describe the package layout and its most relevant member files in a general fashion. Closer examination of each files functions will take place in the following subsections.

```
readme.md
requirements.txt
filter/filter.py
filter/filter_xmlable.py
filter/plugin.xml
cli_tools/naive_k_means.py
cli_tools/hierarchical_k_means.py
tools/plot_ops.py
tools/file_ops.py
tools/nifti_alignment.py
tests/test_suite.py
tests/context.py
```

naive_k_means.py
This Python file includes the naive K-Means implementation and all necessary auxiliary functionalities. The most essential being *distance function for six dimensional datapoints*, *method of initialisation* and a method of saving the processed result.
This file can be run in the command line or via regular import inside a Python shell or file.

hierarchical_k_means.py
This works in a similar fashion as the naive_k_means in that it can be run from the command line, as well as any Python environment.

plot_ops.py
Provides a vast range of supporting functions, necessary for drawing the actual visualisation inside Paraview. This is the main module used by the programmable filter as well as its XML counterpart. It consists of a set of functions for preparatory calculations enabling the visualisation and another set to handle all VTK-related operations like drawing points, lines and organising output data for the Paraview pipeline.

file_ops.py
This module provides a set of auxiliary functions necessary to handle the input files holding connection information. The *assemble_edges* method creates a Numpy array holding the 3D-coordinates for every connexel (set of two voxel coordinates) by combining information of the volume information and the connection data as described in the following section. The Numpy array should be saved and can easily be loaded inside the Paraview filter or during precalculation.
This module requires the Python packages Nibabel and CSV to handle the corresponding file formats.

nifti_alignment.py
In the rare and very avoidable case that volume information and mesh are not aligned, this script can serve as a helper to match a set of volume coordinates with a mesh of the brain.

filter.py
A version of the Paraview filter, that has to be added to a "programmable filter" instance's script property in the Paraview hierarchy. Every attribute of the script has to be manipulated in code, most importantly paths to load data will need to be changed.

filter_xmlable.py
This filter is functionally the same as the one above, but it has been prepared for the automatic generation of an xml variant, using the script provided by Paraview developers.

plugin.xml
This is a version of the filter, that can be loaded using Paraviews plugin manager. It comes with a user interface and is the central piece of software this thesis revolves around. The Python source integrated here can be found in this file: filter_xmlable.py
Due to format reduction in this XML file, readability of the Python source is impaired so for any review purposes the Python file should be referred to.

6.2 Application and Workflow

Using the the tools listed above users are able to perform each of the following steps in order to create a visualisation using Paraview. Note that some additional steps involving software not included in this package are necessary.

General setup:
The following instructions will set up a Paraview environment, which allows the live calculation and visualisation of a clustered set of connectivity data in 3D-space. Note that live calculations tend to get slow depending on the system in use. The main

purpose of the software package remains the loading of preprocessed, i.e. clustered connectivity data. This particular workflow will be described in the section following this one.

These steps require external software sources.

1. **Install Paraview** from https://www.paraview.org/download. Version 5.4 has been used during development, but any version above 4.6 should be a viable option to use the software package.

2. **Install dependencies** using *pip*, the Python package management tool. On both macOS and Linux one simple terminal command should suffice:

   ```
   # cd path/to/_brain_viz
   pip install -r requirements.txt
   ```

   This will install the packages Numpy, CSV, NiBabel and Python-VTK as required.

The following steps build entirely upon the software package's contents.

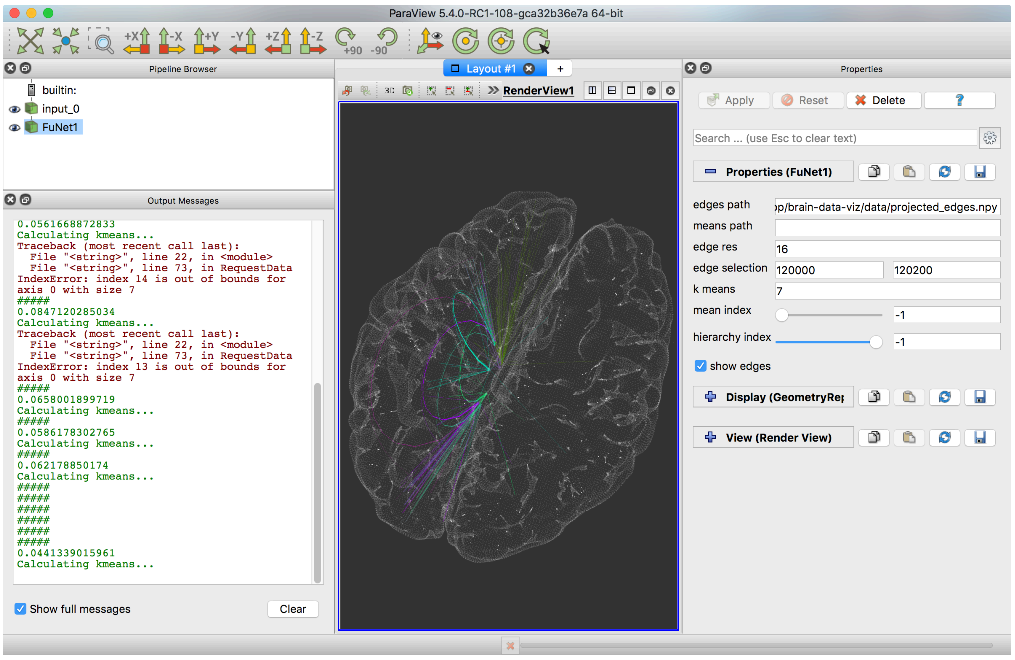
3. **Prepare the Connection Data** using the *file_ ops* module. The simplest way to do this is by importing the module inside a Python shell and saving the edge information to an *.npy* file. The module itself loads Numpy, allowing users to save the assembled edge information from inside it as seen in the last command of the following shell input example.

   ```
   >>> from _tools import file_ops as f
   >>> edges = f.assemble_edges('path/to/.nii', 'path/to/.csv')
   >>> f.np.save('path/to/edges', edges)
   ```

4. **Setup Paraview** by launching the binary inside the downloaded *.zip* file. Load the brain mesh you want to work with by dragging the file to the Pipeline Browser inside Paraview **Fig.6.1**. Some unnecessary display options will be enabled by default and should be disabled for an optimal working experience. **Fig.x, Paraview window.** The Pipeline Browser can be seen on the top left.

5. **Load the XML plugin** using Paraviews plugin manager under *Tools > Manage Plugins... > Load New...*
   The plugin should now appear in the alphabetical list of filters.

6. **Setting up the filter** will require a few extra steps. Most exposed UI attributes have a default value, that can be changed around as desired - however it is absolutely necessary for a user to specify the two fields *bdv path* and *edges path*



```
0.0561668872833
Calculating kmeans...
Traceback (most recent call last):
  File "<string>", line 22, in <module>
  File "<string>", line 73, in RequestData
IndexError: index 14 is out of bounds for
axis 0 with size 7
#####
0.0847120285034
Calculating kmeans...
Traceback (most recent call last):
  File "<string>", line 22, in <module>
  File "<string>", line 73, in RequestData
IndexError: index 13 is out of bounds for
axis 0 with size 7
#####
0.0658001899719
Calculating kmeans...
#####
0.0586178302765
Calculating kmeans...
#####
0.062178850174
Calculating kmeans...
#####
#####
#####
#####
#####
#####
0.0441339015961
Calculating kmeans...
```

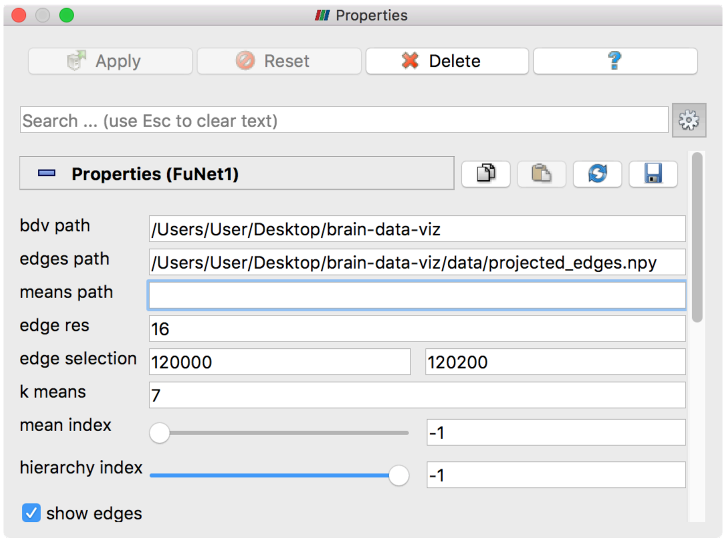
✓ Show full messages

edges path	›p/brain-data-viz/data/projected_edges.npy	
means path		
edge res	16	
edge selection	120000	120200
k means	7	
mean index		-1
hierarchy index		-1

✓ show edges

Fig.6.1: Paraview window. The pipeline browser can be seen on the top left.

according to their personal setup. This is only possible after the filter has been applied to the previously loaded mesh inside the Paraview hierarchy. As the name suggests the *edges path* attribute should be set to the path of the previously created, congregated edge information from step 3. *bdv path* on the other hand should point to the software packages root directory, since it will be used for several imports in the filter source **Fig.6.2**.¶



bdv path	/Users/User/Desktop/brain-data-viz	
edges path	/Users/User/Desktop/brain-data-viz/data/projected_edges.npy	
means path		
edge res	16	
edge selection	120000	120200
k means	7	
mean index		-1
hierarchy index		-1

✓ show edges

Fig.6.2: Properties Panel of the programmable filter. The gear symbol indicates that the advanced view is enabled. This exposes the 'bdv path' property, a path pointing to the root of the software package.

7. Conclusion

7.1 Discussion of Results

The exploratory work done during this thesis was generally successful in the goals it set out to achieve. The first of these goals was the display of curves in 3D-space. A method was introduced, prototyped, abandoned and replaced with a different approach, the Rodrigues Rotation formula, that eventually yielded the expected results. Connexels, consisting of two spatially segregated 3D-coordinates could be visualised by a set of rotated vectors using the VTK pipeline. This was only a setup to the visualisation of whole-scale functional connectivity networks.

In the next step K-Means was introduced as a method of clustering the large number of edges for later, visual summary. A naive K-Means implementation was successfully adapted to be able to perform clustering on a set of six-dimensional connexel datapoints. For this purpose, a distance function was developed, that performed distance computations for a set of edges, each defined by two points in three-dimensional space, without regard to the orientation of said edges. The distance function was only one of the major parts of this specific K-Means implementation. The second one, a barycentre function, was a recomputation of K-Means centroids, which was customised using the same orientation agnostic methods. Continuous tests and benchmarks were applied and proved the functions improvements over regular euclidian distance computations in the context of connexel datasets.

The processing times for large datasets were relatively high, leading to the introduction of a hierarchical implementation of K-Means. This method would create a number of computations of varying numbers of clusters, essentially generating several cluster layouts for the same dataset. The hierarchical scheme would perform K-Means computations consecutively on the centroids, generated by a previous K-Means pass. While not as successful in generating a cluster layout with low aggregated squared distances, this method allowed the exploration of a dataset along the hierarchy offering the possibility of identifying highly expressive cluster distributions for subsets of the data.

The final step towards the visualisation was the integration of the results delivered by the clustering algorithms into the visualisation. The clustering layouts were used to converge edges in 3D-space, creating not only a logical clustering but also a visual bundling of connexel curves with similar properties. The bundling method was implemented in a highly readable manner to expose several attributes of interest to the operator, like the level of compression of bundled edges, as well as the overall convergence of edges with their respective cluster centroid.
Section 5.3 introduced a set of consecutive steps to improve the cluster visualisation by manipulating the exposed features and in that identified two promising methods of visualisation, which hold exclusive values. First was the interpolation of edges,

projected at close-to-constant radius, a visualisation that clearly displayed cluster variance at endpoint neighbourhoods and already reduced screen-space clutter at cluster centres. Second was a method, using varying radii, which yielded a much more condensed visualisation without obscuring endpoint locations, achieving most of the previously stated goals of the visualisation.

At the conclusion of the practical phase, the software package is a solid foundation for visualising functional connectivity data in a number of different ways, that motivate further exploration. K-Means has been successfully implemented to divide any number of connexel edges into any number of clusters and the result is automatically displayed as a set of bundled curves of varying colours (associated with the clusters index).

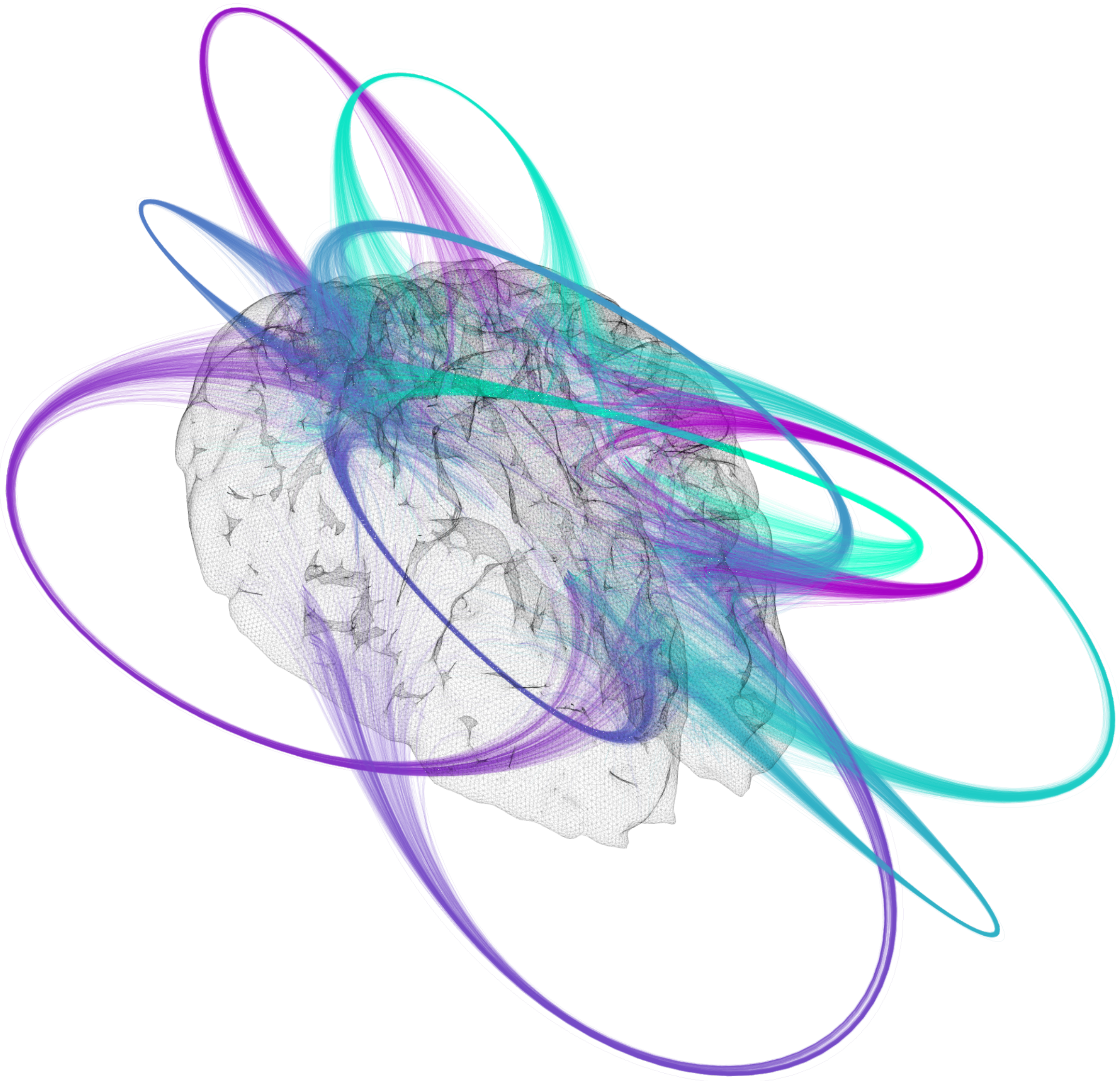


Fig.7.1: externalised connexel curves. n=10.000, k=11

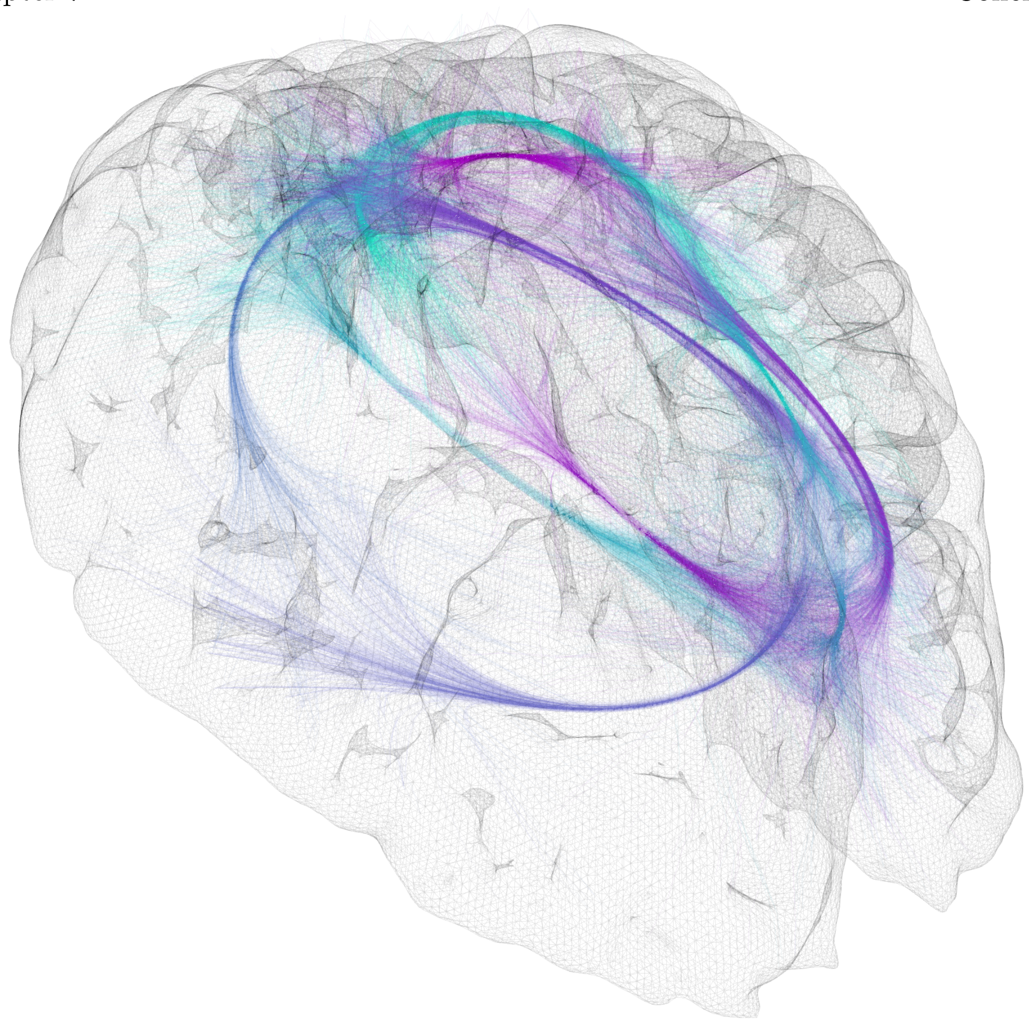


Fig.7.2: internalised connexel curves. n=10.000, k=11

7.2 Outlook

In its current state, the software is able to generate visualisations of functional connectivity data with only a small set of user inputs. Visualisations created for datasets, containing up to 10.000 connexels achieve overall clear visualisations using the default parameters. Larger datasets however, the visualisation of which is the ultimate goal in a project such as this, require more extensive customisations. These customisations are not yet implemented in an accessible enough way, such that only developers with further knowledge of the software will know about them.

Another factor, that can be improved upon is the efficiency of the implemented algorithms. It has been mentioned, that K-Means has been implemented with little regard to efficiency, meaning, that it should eventually be replaced by an implementation with better performance, as well as improved consistency with regard to cluster results. Additionally, the edge bundling method is not performing well and in fact takes even more processing time than the K-Means computation. This can no doubt be fixed in further iterations.

Finally, due to time constraints, the software has not yet been presented to the group of Dr. Johannes Stelzer. Because of the prototypical nature, the discussion should be driven forward by the requirements presented by the functional connectivity researchers, who requested this software. The implemented features will need to be tested by them.

A. References

[Ogawa1990] Ogawa, S., Lee, T. M., Kay, A. R. & Tank, D. W., "Brain magnetic resonance imaging with contrast dependent on blood oxygenation" *Proceedings of the National Academy of Sciences USA 87*, 9868–9872, 1990

[Singleton2009] Singleton, A.B., Camargos, S., Scholz, S., Simon-Sanchez, J., Paisan-Ruiz, C., Lewis, P., Hernandez, D., Ding, J., Gibbs, J.R., Cookson, M.R., Bras, J., Guerreiro, R., Oliveira, C.R., Lees, A., Hardy, J., Cardoso, F. "DYT16, a novel young-onset dystonia-parkinsonism disorder: identification of a segregating mutation in the stress response protein prkra." *Lancet Neurology*, 7, 207-215, 2008

[Nieuwenhuys1998] Nieuwenhuys, R; Donkelaar, HJ; Nicholson, C. "The Central Nervous System of Vertebrates", *Volume 1. Springer.* 11–14, 1998

[Roy1890] Roy, C. S. & Sherrington, C. S., "On the regulation of the blood supply of the brain", *American Journal of Physiology*, 85-108, 1890

[Smith2013] Smith, Stephen M et al. "Resting-State fMRI in the Human Connectome Project.", *NeuroImage 80,* 144–168., 2013

[Dillow2010] Dillow, C. "The Human Connectome Project Is a First-of-its-Kind Map of the Brain's Circuitry", https://www.popsci.com/science/article/2010-09/introducing-human-connectome-project-first-its-kind-map-brains-circuitry

[Biswal1995] Biswal, B., Zerrin Yetkin, F., Haughton, V. M. and Hyde, J. S., "Functional connectivity in the motor cortex of resting human brain using echo-planar", *mri. Magn. Reson. Med.*, 34: 537–541. 1995

[Friston1996] Friston K. J. et al., "Functional Topography: Multidimensional Scaling and Func- tional Connectivity in the Brain", *Cerebral Cortex 60*, 156-164, 1996

[Kimberg2000] Kimberg, D.Y., Aguirre, G.k., D'Esposito, M., "Modulation of task-related neural activity in task-switching: an fMRI study" *Cognitive Brain Research, Volume 10, Issues 1–2*, 189-196, ISSN 0926-6410, https://doi.org/10.1016/S0926-6410(00)00016-1.

[Buckner2008] Buckner, R. L., Andrews-Hanna, J. R. and Schacter, D. L., "The Brain's Default Network". *Annals of the New York Academy of Sciences*, 1124: 1–38. doi: 10.1196/annals.1440.011 2008

[Greicius2003] Greicius, M.D., Krasnow, B., Reiss, A.L., Menon, V., "Functional connectivity in the resting brain: a network analysis of the default mode hypothesis", *Proceedings of the National Academy of Sciences in the United States of America, vol. 100:* 253-258, 2003

[vandeVen2004] van de Ven, V. G., Formisano, E., Prvulovic, D., Roeder, C. H. and Linden, D. E.J. (2004), "Functional connectivity as revealed by spatial independent component analysis of fMRI measurements during rest." *Hum. Brain Ma*, *22*: 165–178. doi:10.1002/hbm.20022

[Worsley1998] Worsley, K. J., Cao, J., Paus, T., Petrides, M. and Evans, A.C. (1998), "Applications of random field theory to functional connectivity." *Hum. Brain Ma, 6*: 364–367. doi:10.1002/(SICI)1097-0193(1998)6:5/6<364::AID-HBM6>3.0.CO;2-T

[Achard2006] Achard, S., Salvador, R., Whitcher, B., Suckling, J., Bullmore, E., "A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs", *Journal of Neuroscience, 4*: 63-72, 2006

[Boettger14] Böttger J., Vilringer A., Schäfer A., Margulies D. S. & Lohmann G., “Three-Dimensional Mean-Shift Edge Bundling for the Visualization of Functional Connec- tivity”, *IEEE Transactions on visualization and computer graphics*, Vol. 20, No. 3, 471-480, 2014

[Bha05] Bharat Biswal, F. Zerrin Yetkin, Victor M. Haughton and James S. Hyde, “Func- tional connectivity in the motor cortex of resting human brain using echo-planar MRI", *Magnetic Resonance in Medicine,* 537-541, 2005

[Loh16] Lohmann G., Stelzer J., Zuber V., Buschmann T., Margulies D. S., Bartels A. & Scheffler K., “Task-Related Edge Density (TED)—A New Method for Revealing Dynamic Network Formation in fMRI Data of the Human Brain”, https://doi.org/10.1371/journal.pone.0158185, 2016

[Pajevic1999] Pajevic, Sinisa, and Carlo Pierpaoli. "Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: application to white matter fiber tract mapping in the human brain." *Magnetic resonance in medicine* 42.3 (1999): 526-540.

[Margulies2013] Margulies, Daniel S., et al. "Visualizing the human connectome." *NeuroImage* 80 (2013): 445-461.

[Margulies2007], image source, Margulies, Daniel S., et al. "Mapping the functional connectivity of anterior cingulate cortex." *Neuroimage* 37.2 (2007): 579-588.

[Beckmann2005] Beckmann, Christian F., et al. "Investigations into resting-state connectivity using independent component analysis." *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1457 (2005): 1001-1013.

[DeLuca2006] De Luca, M., et al. "fMRI resting state networks define distinct modes of long-distance interactions in the human brain." *Neuroimage* 29.4 (2006): 1359-1367.

[Venkataraman2009] Venkataraman, Archana, et al. "Exploring functional connectivity in fMRI via clustering." Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on. IEEE, 2009.

[Lee2012] Lee, Kyu-Min, et al. "Correlated multiplexity and connectivity of multiplex random networks." *New Journal of Physics* 14.3 (2012): 033027.

[Shirer2011] Shirer, W. R., et al. "Decoding subject-driven cognitive states with whole-brain connectivity patterns." *Cerebral cortex* 22.1 (2012): 158-165.

[Honey2007] Honey, Christopher J., et al. "Network structure of cerebral cortex shapes functional connectivity on multiple time scales." *Proceedings of the National Academy of Sciences* 104.24 (2007): 10240-10245.

[Hutchison2015], image source, Hutchison, R. Matthew, et al. "Dynamic functional connectivity: promise, issues, and interpretations." *Neuroimage* 80 (2013): 360-378.

[vanHorn2012], image source, Van Horn, John Darrell, et al. "Mapping connectivity damage in the case of Phineas Gage." *PloS one* 7.5 (2012): e37454.

[Holten2006] Holten, Danny. "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data." *IEEE Transactions on visualization and computer graphics* 12.5 (2006): 741-748.

[McGonigle2011] Schwarz, Adam J., and John McGonigle. "Negative edges and soft thresholding in complex network analysis of resting state functional connectivity data." *Neuroimage* 55.3 (2011): 1132-1146.

[Foucher2005] Foucher, J. R., et al. "Functional integration in schizophrenia: too little or too much? Preliminary results on fMRI data." *Neuroimage* 26.2 (2005): 374-388.

[Gre09] Greicius M. D. & Damoiseaux J. S., "Greater than the sum of its parts: a review of studies combining structural connectivity and resting-state functional connectivity", *Brain Structure & Function,* Vol. 213, Issue 6, 525-533, 2009

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Osnabrück, den 01.11.2017 ....