Imperial College of Science, Technology and Medicine

Department of Computing

MEng Computing (Games, Vision and Interaction)

The Cortical Explorer: A Web-based user-interface for the exploration of brain data

Individual Project Final Report

Author:

Samuel Budd

sfb113

CID:

00846033

Project Supervisor:

Dr. Bernhard Kainz

Second Marker:

Dr. Emma Robinson

June 19, 2017

Abstract

Recent advances in surface based analysis of the brain have resulted in a ground-breaking parcellation (area map) of the brain to be produced. The study that produced this parcellation details the subdivision of the brain's surface into 180 regions per hemisphere, each with different functions and micro-structure. A framework from which neuroscientists can compare different brains is a vital component of neuroscience and this can be achieved using a parcellation of the brain: defining every brain as being composed of a number of different regions.

A challenge currently faced by those that promote surface based brain analysis is exposing the data to other neuro-scientists and the public. Brain data is complex and non-intuitive to work with, acting as a significant barrier to the development of neuro-imaging technologies as the constraints and limitations of the data are rarely understand . The brain and neuroscience is a topic of great interest to the public, however few tools exist to make this highly complex data accessible enough to capture that interest.

This project is an effort to solve these problems by building a set of tools that expose this complex data in an accessible environment (specifically to explore the parcellation of the brain). Meta-data and supporting information were extracted and novel approaches to exploring this data in a interactive environment were developed.

The outcome of this effort was the Cortical Explorer, a web-based user-interface for the exploration of brain data. This allows users to explore the parcellation of the brain in a 3D scene. This facilitates easy access to the supporting parcel meta-data through intuitive interaction

with individual parcels of the brain. Produced alongside this were a set of data processing tools that convert CIFTI files storing complex brain data to VTK files that store a 3D model of the brain in a format suitable for visualisation on the web. This VTK file can be used to generate individual parcel mesh models using this data processing framework.

Acknowledgements

Dr. Bernhard Kainz, Dr. Emma Robinson, The Human Connectome Project, Imperial College London, Glasser MF, Van Essen D, Coalson TS, Robinson EC, Hacker CD, Harwell J, Yacoub E, Ugurbil K, Andersson J, Beckmann CF, Jenkinson M, Smith SM.

Contents

1 Introduction and Motivation

1.1 Motivation

The Human Brain is the most powerful computer in the world, but understanding it is still a challenge we are yet to overcome. Recent advances in neuro-imaging have made it feasible to examine human brain connectivity systematically and across the whole brain in large numbers of individual subjects. The Human Connectome Project[16] have produced a large and accomplished set of high quality neural data, with the objective of studying human brain connectivity and its variability in healthy adults[121]. Glasser et al have used this unique collection of data to extract the most accomplished parcellation of the brain to date, splitting the brain into 180 distinct cortical areas per hemisphere[72]. Parcellation is the process of subdividing the brain's cortical surface into anatomically or functionally distinct regions. A parcellation forms a map of identifiable areas of the brain. The parcellation used in this project is shown in Figure 1, this was provided by Glasser et al[72]. This parcellation was generated using 'Multi-modal' approach meaning that results of multiple imaging modalities were combined to generate the parcellation, results were obtained for a group of subjects and averaged to obtain a single parcellation.

This parcellation of the brain is a powerful resource for furthering our understanding of how the human brain functions. The goal is to identify specialised regions in the brain, these regions are connected to each other in a network, or 'connectome'. Complex functions of the brain are underpinned by the communications sent through this network. Understanding how these

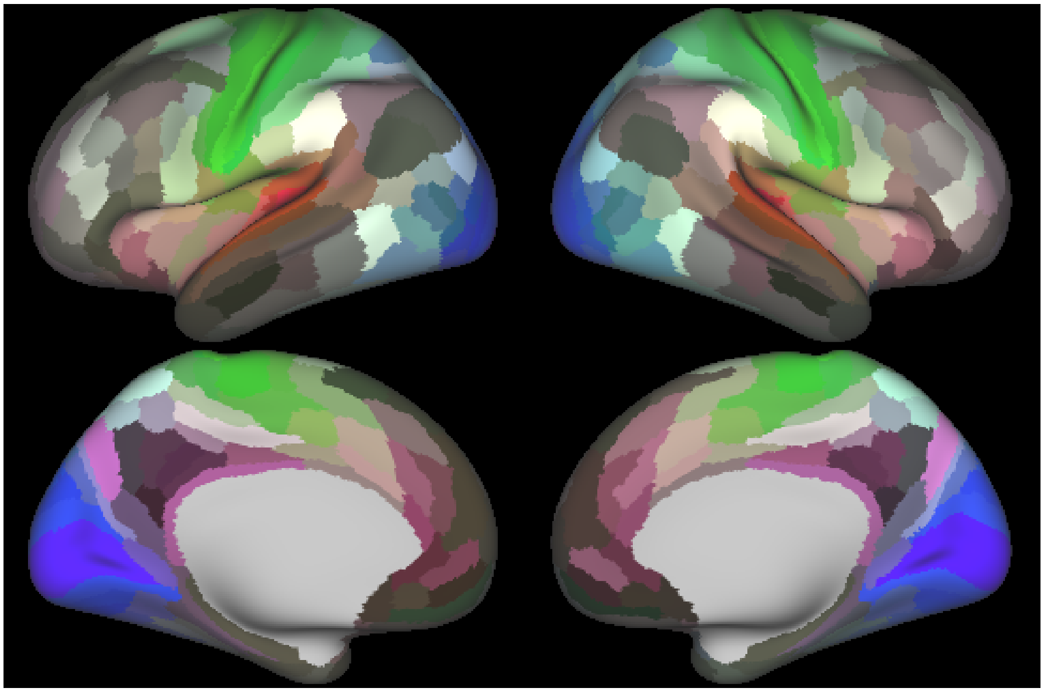


Figure 1: Glasser et al's parcellation of the brain[72]

regions and their connections differ between subjects is an important aspect of understanding the effects of ageing, neuro-development and disease. In order to establish differences between subjects it is vital that neuroscientists have a framework through which to compare brains, this is achieved by defining the brain as being composed of distinct regions - i.e. a parcellation of the brain.

The largest challenge faced by advocates for this type of research is that brain data is incredibly complex and non-intuitive to work with. Specialised training is needed to understand what different regions of the cortex do, and how the data relates back to the brain at a cellular level. This creates a significant barrier for the development of neuro-imaging technology as the limitations and constraints of the data are rarely understand by those without special training. The HCP promotes the surface modelling approach (representing data on the brains surface rather than volumetrically) as this has many advantages, as described in the HCP's pipeline paper[73], this is not being incorporated in the research community as extensively as hoped. The neuroscience community traditionally works with volumetric data and there is a steep learning curve associated with processing and visualising surface based data.

The human brain is an area of science with great public appeal, but as of yet no tools exist that present brain data in an accessible enough form to feed that interest. Public engagement and outreach is vital in helping the world become science literate and encouraging children of all backgrounds to engage with the science community.

1.2 Objectives

This project tackles the challenges described above by setting out the following objectives.

1. Present the parcellation and supporting data in an accessible interactive 3D web environment, developing novel ways of exploring the data through interaction with individual parcels.

2. Provide a set of tools to enable conversion of surface brain data to a format suitable for web visualisation.

3. Provide a tool to enable generation of individual parcel models for visualisation on the web.

4. Extracting meta-data for the parcellation and presenting this in an intuitive and easy to interpret form.

5. Provide a compelling and meaningful visualisation of the brain that engages neuroscientists with surface based brain analysis and engages the public with neuroscience.

2 Background and Related work

Section 2.1 discusses some background neuroscience and Glasser et al's parcellation, as well as the supporting data. Section 2.2 discusses existing tools for medical image visualisation and their strengths and weaknesses. Section 2.3 assesses various technologies for visualising interactive 3D content on the web for their suitability for use in this project.

2.1 Neuroscience and application background

2.1.1 The Human Connectome Project

The Human Connectome Project is an ambitious 5-year effort to characterize brain connectivity and function, and their variability in healthy adults [121]. A consortium of ten institutions in the United States and Europe led by Washington University and the University of Minnesota, has studied a population of 1200 subjects (twins and their non-twin siblings) using High Angular Resolution Diffusion Imaging (HARDI) data along with gathering extensive behavioural and genetic data for each subject. A second group led by Harvard/MGH and UCLA have developed an advanced MRI scanner for diffusion imaging. As per the HCP website:

'The Human Connectome Project aims to provide an unparalleled compilation of neural data, an interface to graphically navigate this data and the opportunity to achieve never before realised conclusions about the living human brain.' [16]

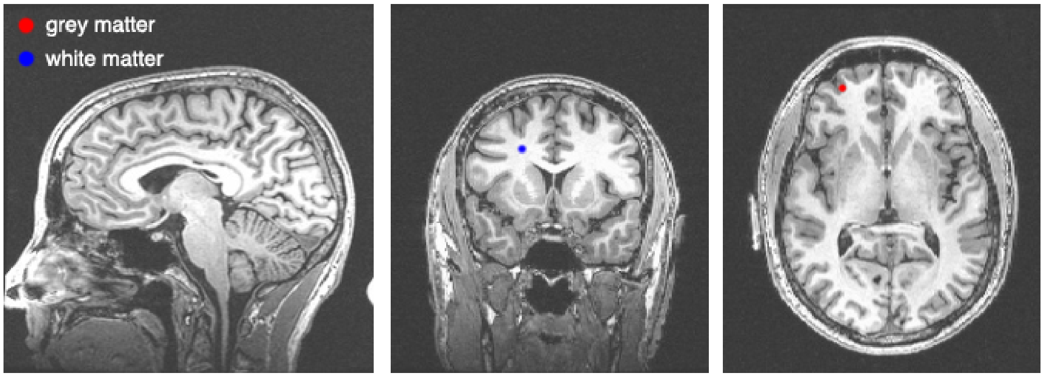


Figure 2: T1 weighted volumetric brain image[1]

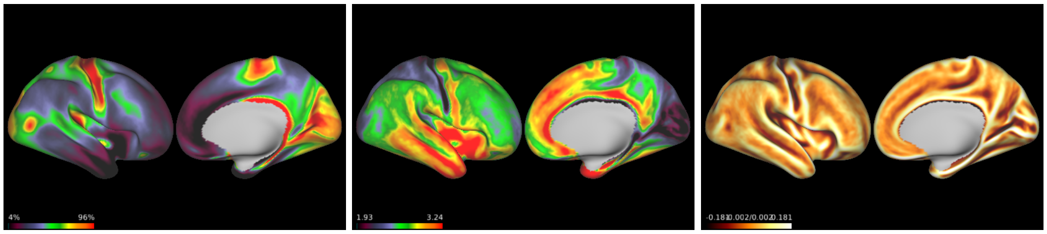
MRI and Imaging Modalities

Magnetic Resonance Imaging is widely used medical imaging technique that provides high quality image data through non-invasive technology. MRI is flexible, allowing study of the brain structure, function, and coarse scale neuronal connectivity.

The Human brain contains an enormous amount of neurons that are distributed among white matter and grey matter. Grey matter has a higher concentration of neurons, and is where the majority of processing takes place. White matter has a lower concentration of neurons but these neurons have longer axons that are used to transmit electrical signals between areas of grey matter.

MRI allows for the study of the brains structure by producing images of the brain's white and grey matter. Figure 2 shows a typical result of an MRI scan (after processing), with white and grey matter annotated. This type of image can be used for structural and myelin mapping as shown in Figure 3

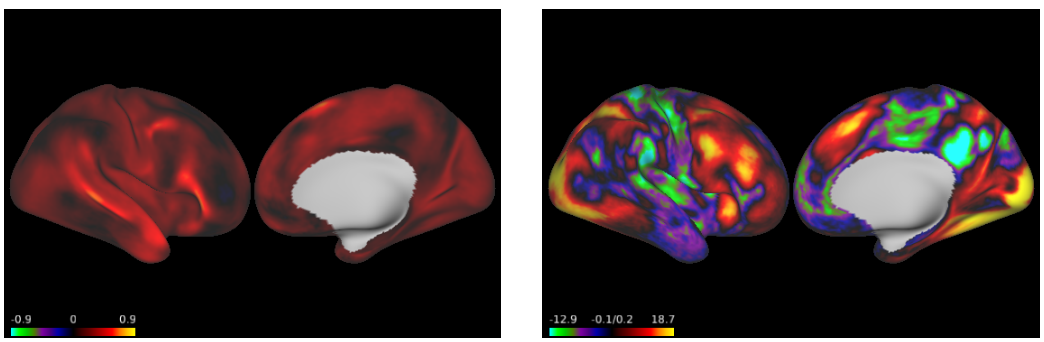
MRI allows for the study of brain function by measuring the activity of



(a) Myelin map (b) Cortical thickness map (c) Cortical Curvature map

Figure 3: Structural MRI data

areas of the brain. Subjects can be at rest (doing/thinking about nothing specific for an hour), producing resting-state fMRI (R-fMRI) , or completing tasks (e.g. reading, listening to music), producing task-evoked fMRI (T-fMRI) as shown in Figure 4.



(a) Resting-state fMRI (b) Task-evoked fMRI

Figure 4: Functional MRI data

MRI allows for the study of course scale neuronal connectivity using diffusion tractography, this extract the white matter connections between grey matter regions the result of this is shown in Figure 5.

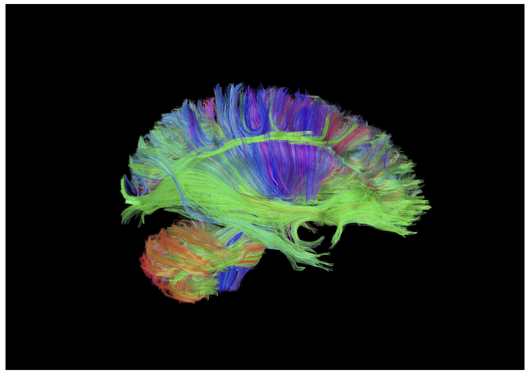


Figure 5: Diffusion Tractography [14]

2.1.2 Multi-Modal Parcellation of the brain

Building an accurate areal map of the brain has been a century old objective in neuroscience[72]. In order for neuroscientists to effectively establish differences in subject a framework must be in place by which brains can be compared. An areal map/parcellation is way of providing this framework. Practical applications of this include comparing populations of healthy and diseased brains which is useful for modelling the mechanisms of disease. A parcellation can be used for brain network studies as well as hypothesis-based focused studies of particular regions of the brain and can be used to standardise the reporting of results.

The high quality data curated by the HCP has allowed new levels of analysis of multi-modal magnetic resonance images. Using the modalities described above, Glasser et al used a semi-automated approach resulting in 180 areas per hemisphere - bounded by significant changes in cortical architecture, function, connectivity and topography[72]. Using the data from 210 healthy young adults, Glasser et al characterized 97 new areas and 83 areas previously reported using study-specific approaches, each of these area's properties were documented and related to existing literature. To take full advantage of this parcellation, a machine learning classifier was trained to recognize the multi-modal 'fingerprint' of each cortical area. This classifier enables automatic parcellation and identification of these areas in new HCP subjects and future studies. The parcellation data has been made freely available in the BALSA database[3].

Figure 1 shows the final parcellation produced by the Glasser et al. Each area is colour coded based on broad categories of function: 'Auditory', 'Visual' and 'Sensory and Motor'.

File Formats

The HCP surface data is saved in two file types:

1. GIFTI (*.gii*): GIFTI files can represent surface mesh files or surface vertex data. Vertex data can be continuous scalar values (e.g. when representing a myelin map) or discrete labels (e.g. when representing a parcellation) and are stored as data arrays. Surface mesh files contain a list of vertices and a list of face triplets[79].

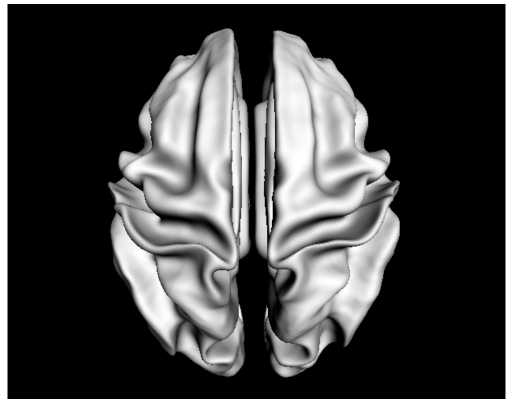
2. CIFTI (*.nii*): The main difference between CIFTI and GIFTI is that GIFTI files only represent surface information for one hemisphere of the brain, whereas CIFTI files represent volume data for both hemispheres.

At present it is possible to visualise and manipulate these data types using various applications discussed in Section 2.2.

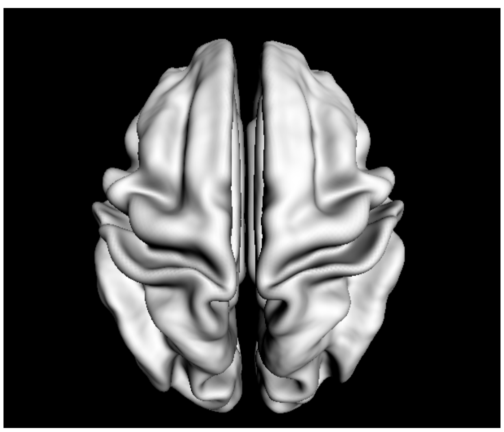
Cortical Surface Representations

The HCP provide several surface models. Figure 6 shows each of the available surface files. Figures 6a, 6e and 6f are used for visualisation purposes and do not reflect the surface of any part of the brain. Figure 6b shows the 'White' surface which is the boundary between white and grey matter (See Figure 2). Figure 6d shows the 'Pial' surface which is the outer surface of the brain. Figure 6c shows the 'Midthickness' surface which is halfway between 'Pial' and 'White'.

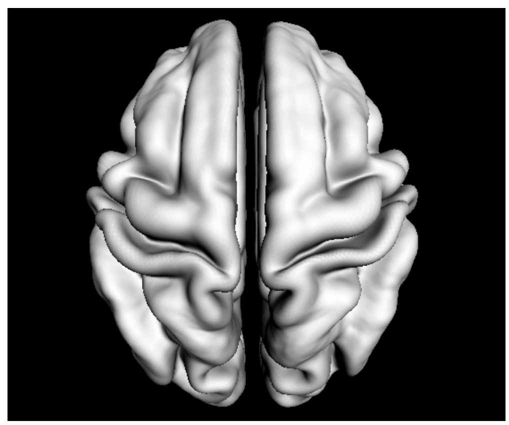
(a) Flat model



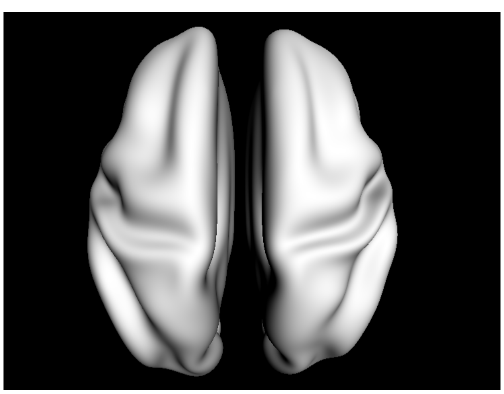
(b) White matter model



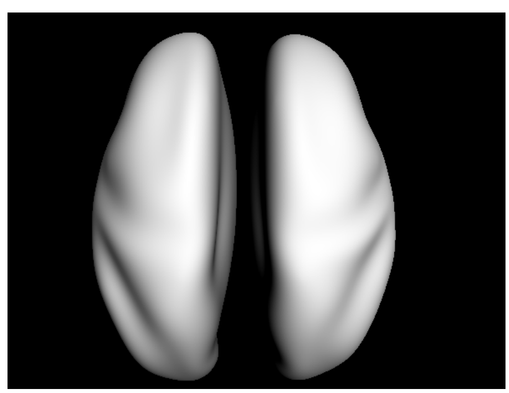
(c) Mid-Thickness model



(d) Pial model



(e) Inflated model



(f) Very Inflated models

Figure 6: The various GIFTI surface models provided (Screen-shots of surfaces visualised in ThreeJS web-page)

2.2 Medical Image Visualisation Techniques

Visualisation can be categorized into two main areas: Scientific Visualisation, and Information Visualisation with the following definitions [94]:

- Scientific Visualisation: the use of interactive visual representations of scientific data, typically physics based, to amplify cognition.

- Information Visualisation: the use of interactive visual representations of abstract, non-physically based data to amplify cognition.

This project centred around Scientific Visualisation, but will leverage the advantages of Information Visualisation to build a compelling and informative visualisation. There are many tools available for visualising medical data. These tools vary in the data types they specialise in and the platforms they support. These tools are discussed in more detail below.

2.2.1 HCP Workbench and Caret

The Human Connectome Project has a dedicated application, the Connectome Workbench[5], for exploring the data they have collected. It is surface and volume visualisation platform developed by the HCP for viewing the many modalities of MRI-based data generated by the HCP's pipelines. It supports standard neuro-imaging NIFTI, GIFTI and CIFTI formats [15]. The Workbench has been built as an extended version of Caret5, a similar neuro-imaging tool[4]. These tools offer many features for both analysis and visualisation. It is a custom application built in C++ by the Van Essen Labs. This tool is the most accomplished and straightforward tool available to view the HCP data. The flaw of this tool is that the parcel meta-data is not accessible within the Connectome Workbench. The Workbench is useful for manipulating the users data but does not explain the results or allow users to answer questions such as 'what is this parcel? what is its function?'. Provided alongside this is a command line utility called wb_command, this

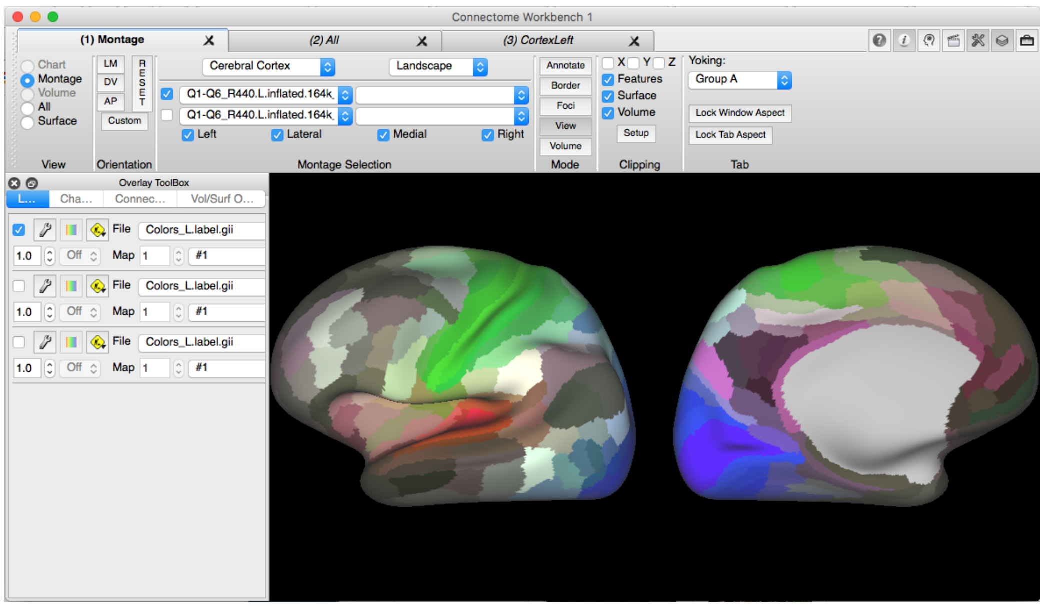


Figure 7: Example Visualisation using the Connectome Workbench (Screenshot)

can be used to manipulate and perform various analyses on data supported by HCP[47].

2.2.2 VTK and ParaView

The Visualisation Tool-kit (VTK) is an open-source software system for 3D graphics, image processing and visualisation. It is a C++ class library with several interpreted interface layers including Java and Python. It supports a wide range of visualisation algorithms, advanced modelling techniques and visualisation frameworks[40].

ParaView is a GUI for building visualisations and analysis using qualitative and quantitative techniques. It uses VTK as the data processing and rendering engine[30].

Although an accomplished visualisation of the brain could be achieved using ParaView, this would be offline and have less flexibility than the Cortical Explorer.

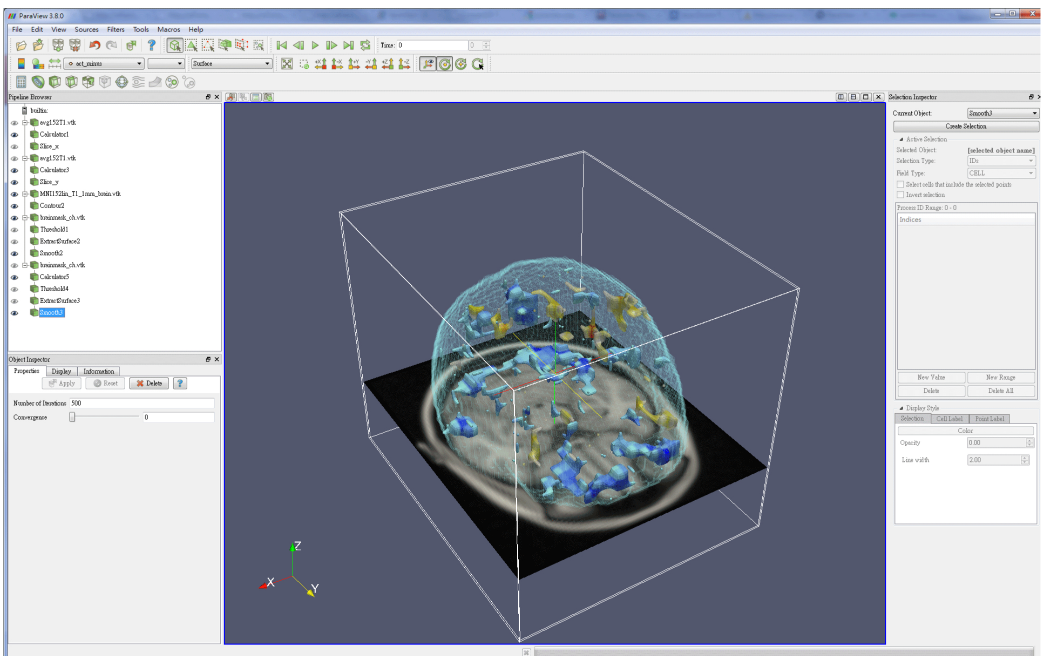


Figure 8: Paraview Application [31]

2.2.3 PySurfer, FreeSurfer and MayaVi

FreeSurfer is a set of software tools for studying cortical and sub-cortical anatomy[10]. It provides a pipeline for extracting cortical surface mesh models from MRI data including labelling of regions on the cortical surface and well as sub-cortical brain structures. It facilitates the visualisation of regions of the brain and contains tools to conduct volume and surface based analyses. This tool also uses VTK to aid in visualisation. FreeSurfer allows users to select a brain surface model to use and choose a metric overlay, however it does not support CIFTI files. Again this tool is off-line so unsuitable as a solution to our problem.

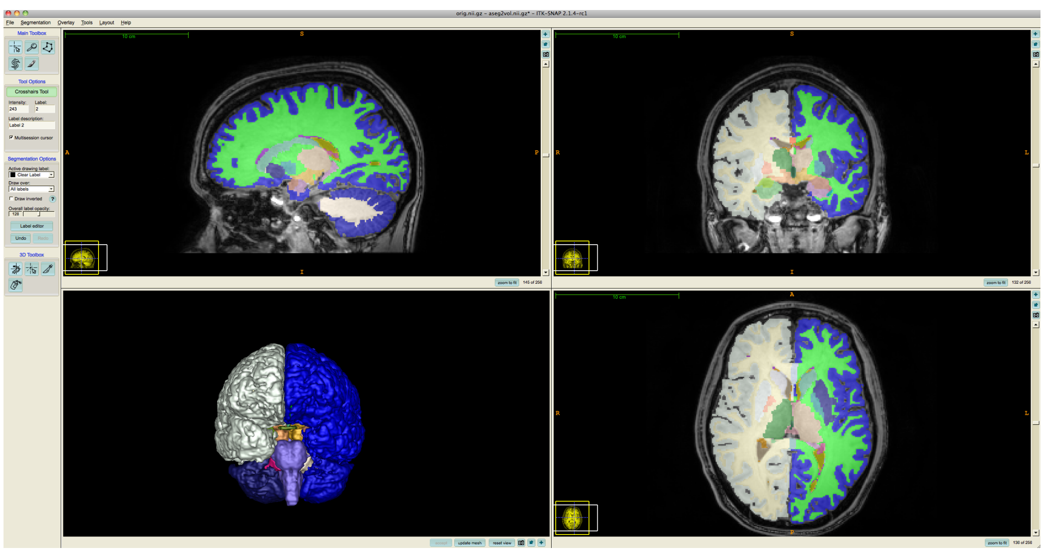


Figure 9: Example Brain data analysis using FreeSurfer [11]

2.2.4 NeuroVault, MangoPapaya and PyCortex

NeuroVault is an on-line repository that allows researchers to store, share, visualize and decode statistical maps of the human brain[74]. It uses the Neurosynth[25] database to process the data uploaded for their purposes. Designed to provide intuitive, interactive visualisation of uploaded images, Neurovault has an embedded JavaScript 2D/3D viewer using the open-source Javascript Papaya/PyCortex libraries respectively. Users can adjust statistical thresholds, select different colour maps, and load additional brain volumes for comparison among other features. Users can interrogate the data in both volumetric space and on the surface. NeuroVault also exposes an API (Application Programming Interface) which could potentially be used to access more data to view within the Cortical Explorer.

Mango (short for Multi-image Analysis GUI) is a viewer for medical research images, providing analysis tools and a user interface to navigate image volumes[20]. Papaya is a Browser based version. It supports NIFTI(2), GIFTI and VTK image and surface formats, unfortunately it does not support CIFTI. An advantage of this application is the surface rendering features which allow interactive surface models supporting cut planes and overlays.

PyCortex and its 3D web viewer allow users to visualise surface maps with anatomical and functional information projected onto the cortical surface, this surface can be inflated and flattened interactively aiding interpretation[66]. It is a similar application to what the Cortical Explorer will be, but there are some areas where Cortical Explorer will be superior, such as interacting with the individual areas found in the parcellation. PyCortex does not work on iPad, and does not support CIFTI files at present. A key let down is the quality of the visualisation is not of a high standard.

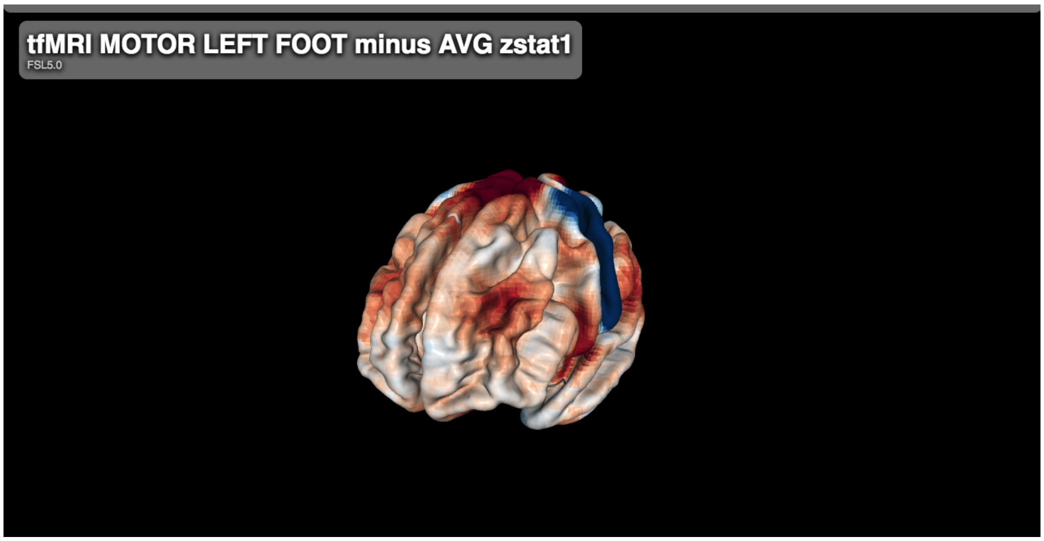


Figure 10: Neurovault's 3D Viewer built with PyCortex [26]

2.3 Interactive 3D Visualisation on the web

There are many different ways of presenting graphics on the web, presenting interactive 3D graphics is more challenging task. After researching various options for visualising 3D content on the web I decided using WebGL (Web Graphics Library[41]) would suit my project as it offers a lot of flexibility and offers among the best performance of all the options[64]. WebGL is a JavaScript API for rendering interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins, it uses the same structure as OpenGL but in a web context, rendering to a canvas element in a html document. In the interest of performance it utilises the devices GPU (Graphics Processing Unit) to accelerate execution time[41]. For the visualisation to be considered interactive, it should respond in real-time to user input, smoothly and fast on all modern devices - WebGL is capable of doing this. For real-time interaction the application should run at 30 Frames-per-second (A good rate for human visual perception) and aim for 60 Frames-per-second (The detectable limit for human visual perception)[13]. One of the main benefits of WebGL is the multitude of frameworks available to streamline work-flow.

There are three options for programming in WebGL, the first is using WebGL directly, the second is using a Javascript API like THREE.js that abstracts away many complexities of WebGL, and the third is using a game engine such as Unity3D which offers the ability to export to WebGL.

Many applications for developing in WebGL offer support to embed visualisation within their own environment, such as Blend4Web [38]. This is not suitable for my application as fine grain control over the supporting front and back ends of the application is required. Research suggested that the following frameworks would be optimal for building my application:

- BabylonJS[2] - "A complete JavaScript framework for building 3D games with HTML5, WebGL and Web Audio"
- ThreeJS[34] - "Three.js is a cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL."
- WhiteStormJS[46] - "Whitestorm.js is a 3D JavaScript library/API based on Three.js that simplify code, adds physics and post-effects. The source code is hosted in a repository on GitHub."
- Unity3D[37] - "Unity is a cross-platform game engine developed by

Unity Technologies and used to develop video games for PC, consoles, mobile devices and websites." (Unity allows some basic interaction between the embedded WebGL section and surrounding website, but I was unsure whether this extended to introducing new models so I left it in for now)

Table 1 shows the core requirements for the project and whether each option supports that requirement.

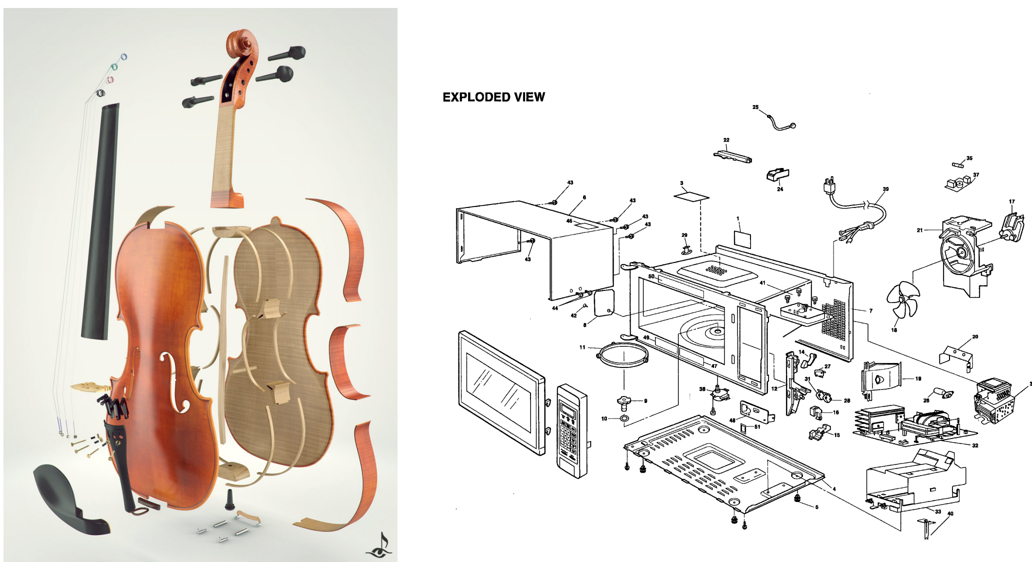
Requirement	WhiteStormJS	BabylonJS	Unity3D	ThreeJS
Free	Yes	Yes	Yes*	Yes
Support	Little	Some	Lots	Lots
Performance	Good	Good	OK	Good
Customisable	Yes	Yes	No	Yes
Lightweight	Yes	Yes	No	Yes
VTK support	No	No	No	Yes

Table 1: Core Requirements (* Free for basic usage, not for my requirements)

2.4 Core Features Background

2.4.1 Explosion Diagram

A key objective of this project is to view an 'explosion' diagram of the brain parcels. Examples of explosion diagrams can be seen in Figure 11. Kerbl et al have created interactive explosion diagrams for complicated 3D objects made up of many components[83]. This is done by organising the 3D components into a hierarchical structure and defining how each object 'explodes' with respect to its parent in the hierarchy.



(a) Explosion diagram of a violin [6]

(b) Explosion diagram of a microwave [7]

Figure 11: Explosion diagram examples

2.4.2 Smart Labelling

A key objective of this project is to implement a labelling algorithm to annotate the parcels of the brain. Tatzgern at al show that real-time 3D labels in a scene can be updated in realtime to remove occlusions between them. They approach this in two different ways[113]:

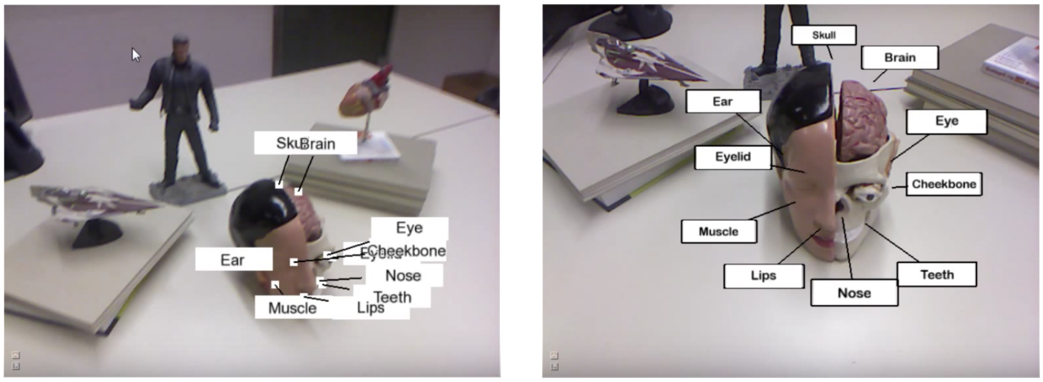
1. Pole based constraints: The label is constrained to only move along a pole that connects the label and the object, the label move up or down the pole to a point where no occlusions with other labels occur. The positions are updated in real time as the viewpoint on the 3D scene changes.

2. Plane based constraints: Each label is assigned to a plane perpendicular to the viewing direction, labels are optimised within each plane to reduce occlusions.

Figure 12 shows an example of naive labelling, and Tatzgern et al's solution that removes occlusions.

2.4.3 VTK Format

The Visualisation Tool-kit (VTK) is a free, open-source software system for 3D computer graphics, image processing and visualisation[40]. VTK is a C++ class library with a Python interface layer (Java is also available). It is designed to be language and platform agnostic. The main advantage of using VTK as a file format is that the data model is designed to be able to represent almost any real-world problem[9]. VTK allows multiple sets of



(a) Example of naive labelling technique[113]

(b) Example of optimised labelling strategy[113]

Figure 12: Labelling techniques shown by Tatzgern et al[113]

per-vertex data to be included in one file. VTK is a supported file format of ThreeJS. VTK offers modelling algorithms that combine various filters that manipulate data geometry. Filters inspect the datasets it is given to produce derived data for output. Connecting these filters creates a data flow network. Compared to other 3D object formats, VTK offers a lot of flexibility and is the only format that is both suitable for visualisation with ThreeJS and backed by a sophisticated set of tools for manipulating this data programmatically. For these reasons VTK was chosen as the format to convert CIFTI files into, and to generate the individual parcel models using VTK's supporting tools.

3 Method

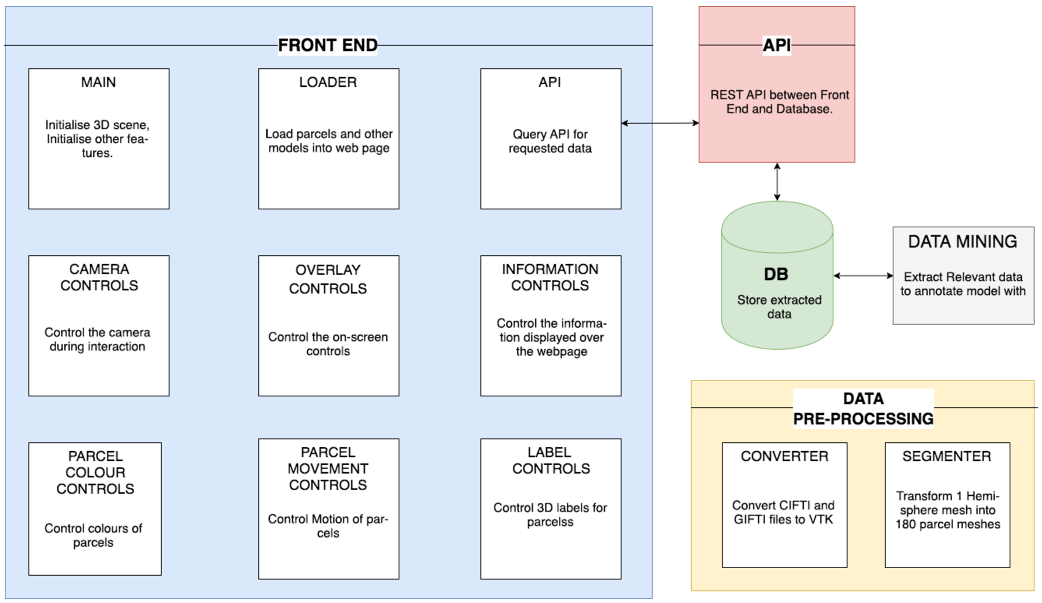


Figure 13: Overview of Proposed system

This section outlines the theory behind how each project objective is achieved. Section 3.1 describes the technology stack used for ’Front-End’, ’API’ and ’DB’ as shown in Figure 13. Section 3.2 describes converting the provided data into a suitable format for visualisation on the web (’Data Pre-Processing’ in Figure 13). Section 3.3 describes an algorithm to generate 360 individual parcel models for one VTK model. Section 3.4 details the components that make up ’Front-End’ as shown in Figure 13. Concrete implementation of the methods desrcibed along with images of the results of each step can be found in Section 4

3.1 Technology stack

This section discusses the steps taken to choose a WebGL framework and the supporting web technologies to develop the web application: The Cortical Explorer.

3.1.1 Prototyping

Alongside research into different 3D web frameworks (Section 2.3), small prototype applications were built in two different frameworks to assess their performance and ease of development. A simple example with 3 spheres that could be interacted with in a similar way to how a final model of the brain would be interacted with was implemented. The two features are moving around/orbiting the objects, and using the mouse to select objects and displaying this interaction on the screen.

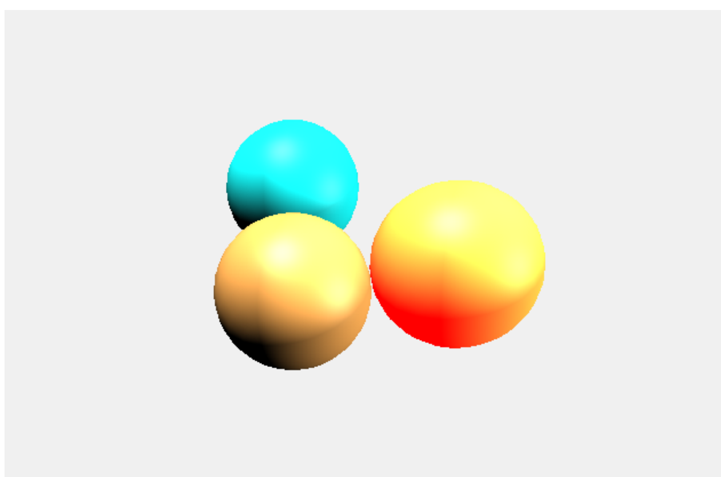
The 3D rendered scene will form the main component of the Cortical Explorer, importantly it has to have the capability to drive its behaviour via instructions from the surrounding application. ThreeJS and Unity3D were chosen to built prototypes in.

ThreeJS was chosen over BabylonJS and WhiteStormJS because, despite being similar, ThreeJS is slightly older and offers a larger support network. ThreeJS also offers a VTK object loader, no other framework offered this. WhiteStormJS uses features of ThreeJS but at this point integrating physics was unnecessary and it also has not got all the features of the ThreeJS implemented (Including the VTK object loader).

ThreeJS

ThreeJS is a JavasScript API for rendering interactive 2D and 3D graphics inside a HTML canvas element. It is supported by all major browsers and works well on mobile and tablet. ThreeJS provides easy access and fine grained control over 'Scenes', 'Cameras', 'Geometry', '3D Model Loaders', 'Lights', 'Materials', 'Shaders', 'Particles', 'Animation' and 'Math Utilities'. ThreeJS provides lots of examples and tutorials to get started so it was straight-forward setting up my prototype scene. Figure 14 shows a screen-shot of the prototype with the mouse hovering over one of the spheres to change its colour.

Figure 14: ThreeJS screen-shot

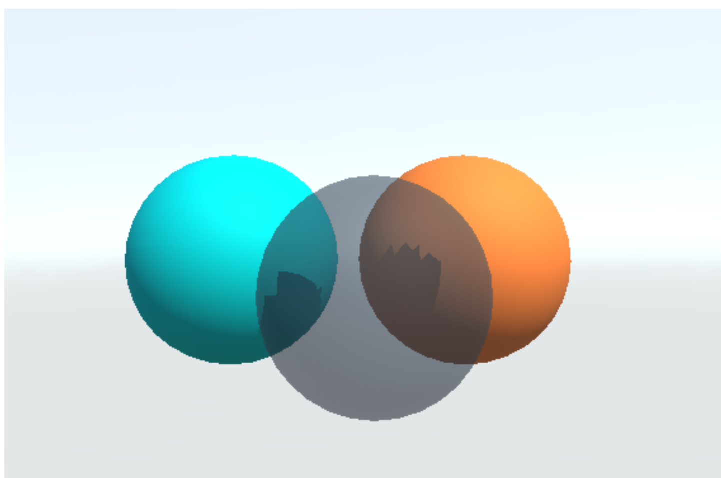


Unity3D

To see how the development process differed between using a direct API and using a game engine, a prototype was built in Unity3D as it came the closest to satisfying the requirements among the different game engines.

Unity3D is a multi-platform game development tool that offers the ability to export applications to WebGL/ HTML[36]. It provides a customizable and easy to use editor. In the Unity editor some simple objects were added to the scene, and then some scripts attached to these objects for mouse orbit controls and interaction with each object. Figure 15 shows a screen-shot of the of the prototype with the mouse hovering over one of the spheres to change its colour.

Figure 15: Unity3D screen-shot



3.1.2 Deciding a WebGL framework

ThreeJS has some key advantages over Unity3D. Firstly the file sizes of the ThreeJS scene are a lot smaller, making it more suitable for use on the web. The files exported by Unity3D are large and take longer to load. ThreeJS performed better on slower devices compared to Unity3D. The shadows in

Unity3D looked good in the editor but once exported to WebGL and viewed on the web, were jagged and don't meet the standards required to achieve the project objectives. Another area Unity3D falls short is that it is not able to handle custom data uploaded by the user, as once the scene is exported to WebGL, no additional models can be added to it. ThreeJS does support uploading custom data to be used within the 3D scene. ThreeJS excels in all the areas examined and it also offers a VTK importer, which was identified as the most suitable format to convert my data into in Section 2.4.3. ThreeJS has all the features and support required to implement each of the features of 'Front-End' shown in Figure 13. It outperformed alternative frameworks on all tested requirements so on balance it was decided that ThreeJS is the best framework to use to achieve my project objectives.

Using a VTK model converted from a GIFTI file (Described in Section 4.1), the VTK loader offered by ThreeJS was tested and was able to produce the results shown in Figure 16.

3.1.3 Supporting Web Technologies

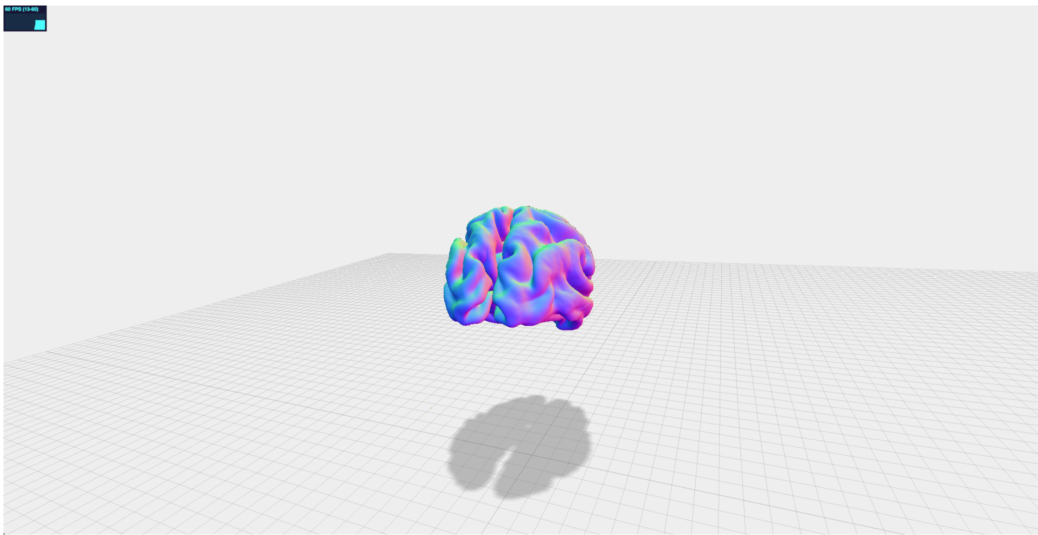
Node.js

As per the Node.js website:

"Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."[27]

Node.js was chosen for both 'Front-End' and 'API' shown in Figure 13 for several reasons:

Figure 16: Testing the VTK loader



- Write both front and back end in Javascript
- Ideal for real-time applications because of non-blocking event-driven I/O
- Compatible with ThreeJS
- Wide ranging documentation, support and tutorials
- Node.js is backed by a vast javascript package library called 'npm'[28] that allows users to implement advanced functionality through simple API's.
- Easily interfaced with MongoDB with 'npm' packages
- Choice of frameworks that aid in user interface design

- Experience using Node.js over alternatives

Ractive.js

Ractive.js is a user-interface library. It is a template-driven framework that aims to streamline building responsive, interactive user interfaces through the use of two-way binding, animations and more[32]. Ractive 'works for you' by not locking you in to a framework-specific approach to development. Ractive.js is the ideal tool to use to build simple on-screen controls as seen in 'Overlay Controls' in 'Front End' of Figure 13

MongoDB

MongoDB is a No-SQL document database solution[45]. MongoDB allows for simple creation of document databases that can store data with a flexible schema, this is important as the data used will likely evolve throughout. Npm offers modules to simplify the interface between the application code and the MongdoDB database. The supporting meta-data for the parcels ('Data Mining' in Figure 13) comes in many different forms which means that the flexibility of MongoDB make it the perfect database to store parcel meta-data in ('DB' in Figure 13).

3.2 Conversion of CIFTI to VTK

This section discusses 'Converter' in 'Data Pre-Processing' as shown in Figure 13. The provided CIFTI and GIFTI files must be converted into a VTK format for two purposes:

- To load into a ThreeJS scene: As previously mentioned in Section 3.1, ThreeJS supports loading VTK models into a scene. VTK models of the different modalities of data are produced for visualisation.

- To segment the hemispheres: To create individual meshes for each parcel in 'Segmenter', shown in 'Data Pre-processing' in Figure 13, a VTK model of each hemisphere showing the parcellation is required.

The main files are:

1. Surface GIFTI files (.surf.gii) - These files contain the vertices and face information needed for the 3D model.

2. Overlay CIFTI files (.dscalar.nii, .dlabel.nii) - These files contain per-vertex data such as myelin maps (per vertex scalar data) or parcel data (per vertex parcel index, dictionary of parcel index to rgba colour).

Contained inside a GIFTI surface file and CIFTI metric file is all the data necessary to produce a VTK model file.

As CIFTI is still a relatively new standard with limited support from existing brain image data manipulation solutions, the Connectome Workbench's command line interface is used to convert the CIFTI files into GIFTI files before processing further.

Converting these GIFTI files into VTK has three core steps as can be seen in Figure 17:

1. $surface = loadSurfaceGIFTI$; - Load the surface GIFTI file, this gives us the surface vertices and faces.

2. $map = loadOverlayGIFTI$; - Load the overlay data, this gives us the per vertex data that is converted into a colour for that vertex.

3. $writetoVTK(surface, map)$; - Manipulate surface and map data to write to a VTK file.

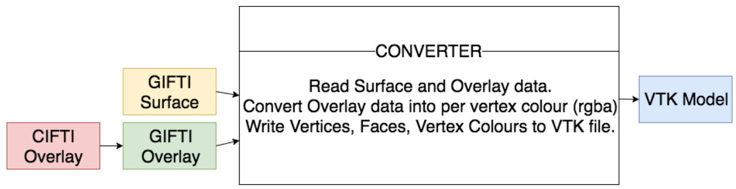


Figure 17: Overview of 'Converter' in 'Data Pre-processing' from Figure 13

3.2.1 Matlab

matlab_GIFTI[21] is an Open Source MATLAB package for handling GIFTI files. This package allows GIFTI files to be loaded into MATLAB. Area label data (*.label.gii*) has the following properties when loaded into Matlab and combined with the surface data:

```
gifti.vertices: [163842x3 single]
gifti.faces: [327680x3 int32]
gifti.cdata: [163842x1 single]
gifti.mat: [4x4 double]
gifti.labels.name: [1x181 cell]
gifti.labels.key: [1x181 double]
gifti.labels.rgba: [181x4 double]
```

Scalar data ($.func.gii$) has the following properties:

```
gifti.vertices: [163842x3 single]
gifti.faces: [327680x3 int32]
gifti.cdata: [163842x1 single]
gifti.mat: [4x4 double]
```

The key advantage of using this package is that it already offers the functionality to write GIFTI files to VTK. The toolbox does this by loading a GIFTI file and checking for what properties the loaded 'GIFTI' object has. It writes vertices and faces to VTK as well as checking for colour data and writing these to the VTK output file only if they are present in the GIFTI object.

The disadvantage of this approach is that this is difficult to bring on-line and restricts users from uploading their own GIFTI files. Section 6 discusses some alternate solutions to this problem.

3.3 One Surface to 360

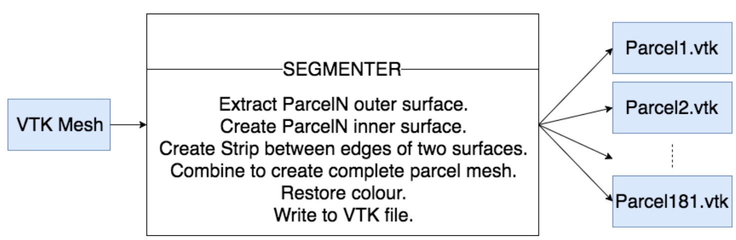


Figure 18: Over view of 'Segmenter' from 'Data Pre-processing' in Figure 13

The VTK file with the correct information stored inside (parcel/area data) for each hemisphere (Output of converting *label.gii* parcellation data to VTK) is split into 181 separate parcel meshes, as shown in 'Segmenter' in

Figure 13. The Python VTK package allows users to build their own processing pipelines[40]. My proposed algorithm for separating each surface into individual parcel volumes, as seen in Figure 18 is as follows:

1. $mesh = loadwhole\_brain\_surface$; - Load the vtk mesh into Python.

2. $parcelouter\_surface = mesh.filter(vertex.parcel\_index = current\_parcel\_index)$; - remove any vertices and faces that are not part of the target parcel. Figure 19 shows the output of this step

3. $parcelinner\_surface = out\_surface.offsetAndShrink(offset\_scale)$ - create a copy of the $outer\_surface$ and translates this surface away from the original to become the $inner\_surface$, it is then shrunk towards its own center to create a tapered edge. Output is shown in Figure 19.

4. $parcelstrip = joinEdges(outer\_surface, inner\_surface)$; - This extracts the edges of $outer\_surface$ and $inner\_surface$ and creates 'strip' between the two, as shown in Figure 19.

5. $parcelcombined = combine(outer\_surface, inner\_surface, strip)$; - This combined the $inner\_surface$, $outer\_surface$ and $strip$ into one vtk model, then the colour of that parcel is restored to cover the whole parcel, the result is shown in Figure 19.

6. $writecombinedtoVTK$; - Output the parcel mesh to a VTK file.

The Parcel meshes are in a format suitable for visualising in ThreeJS after these steps are completed.

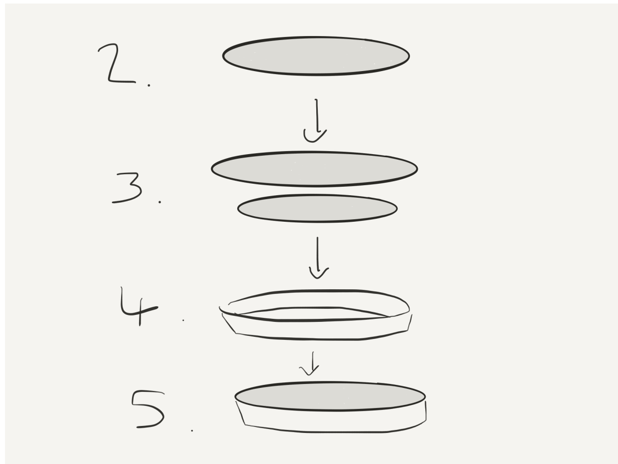


Figure 19: Graphical overview of the steps described for 'Segmenter' as shown in Figure 18

3.4 360 Parcels, Interactive on the web

The models generated during the previous step are the focal point of this application. Discussing User Stories[39] with Dr Kainz, Dr Robinson and Professor David Van Essen, University of Washington, St Louis (Lead PI of the HCP Project) gave a better idea of what the most urgent features are and the key considerations that have to be made. This section discusses the 'Front End' as shown in Figure 13 as well as aspects of 'API' and 'DB'. Figure 20 shows an overview of the user flow for the proposed application.

3.4.1 Basic Interaction

Starting in the top left of Figure 20, the following steps are taken to give users basic interaction with the parcels, this interaction is the same as described in Section 3.1.1.

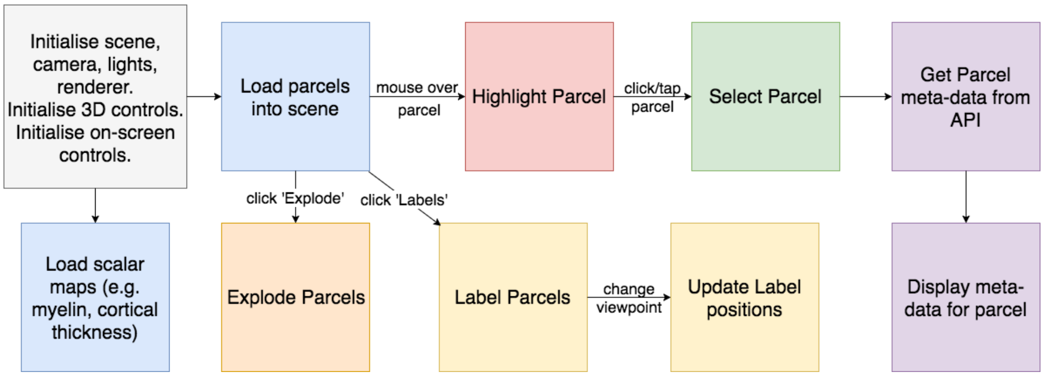


Figure 20: User flow through 'Front End' shown in Figure 13 (Starting in top left)

1. ThreeJS is used to initialise the rendering loop and embed a 3D scene within the web-page. This uses ThreeJS's WebGL 'Renderer'. Ambient light is added to the scene so that users can see the objects in the scene[44]. A 'Spotlight' is also added to the scene so that the depth of objects is discernible and objects appear more realistic to the eye[19]. This step requires a 'Camera' to be placed in the scene to act as our view-point on the scene, this will be a 'Perspective' camera instead of a 'Orthographic' camera by default, scenes renderer using 'Perspective' appear more realistic to the human eye and allows users to see depth in the scene[43].

2. Use ThreeJS's 'Orbit Controls' to initialise interaction with the scene. 'Orbit Controls' supports zooming in/out, rotating camera around 'target' (The centre of the parcels in this case), and panning the scene[29] (These features are device independent). Sensible limits are placed on these controls (e.g. users cannot zoom so far out that they cannot see

the brain).

3. Still in the top left of Figure 20, Ractive is used to initialise on-screen controls that allow users to show/hide hemispheres of the brain by setting each of parcels in that hemisphere to invisible. Buttons to view the brain from common viewpoints (e.g. top, left, front), these update the camera position to the respective point and direction within the scene to view the brain from the chosen face. A button to 'Explode' the parcels described in Section 3.4.4. A toggle for 'Labels' described in Section 3.4.5. A button to 'Inflate' the parcels and buttons to navigate to visualisations of scalar data such as myelin maps, both described in Section 3.4.6. On-screen controls are organised on the screen so that they occlude as little of the brain as possible while still being usable.

4. The 'Load parcels into scene' section of Figure 20 uses ThreeJS's 'VTK-Loader' to load the geometry from each parcel's VTK file. This geometry is then combined with a 'Material' to create a ThreeJS 'Mesh', this 'Material' must use the 'Vertex Colours' from the geometry to have colours consistent with Glasser et al's visualisation. A 'Lambertian' material is chosen to give each parcel a realistic visual effect[85]. The 'inflated' model of the brain will be used to generate the individual parcel meshes as this is the most commonly used surface model for presentation purposes by Glasser et al.

As shown in Figure 20, once the parcels are loaded into the scene, the user has three routes they can take in the application, 'Highlight Parcel' and 'Select Parcel' are discussed next.

3.4.2 Highlighting and Selecting Parcels

When navigating the brain it must be clear to user when they have the option to interact with the parcels. When a user 'hovers' their mouse over a parcel, both change appearance to indicate the a parcel can be 'selected', this user flow can be seen in Figure 20. The 'hover' action is only applicable on desktop environments but concepts from 'highlighting' parcels are used to 'select' parcels as well.

By tracking the position of the mouse on the screen, ThreeJS's 'Raycaster' is used to 'cast' a ray into the scene and return a list of objects that this ray intersects i.e objects that are underneath the mouse in the scene[23]. Intersection is calculated by checking whether the triangles that make up each object intersect with the ray. The first element of the returned list is the object under the mouse that is closest to the camera and the most visible object under the mouse. This object is 'highlighted'.

To highlight the object, the object's 'Emissive' property is updated. The 'Emissive' property lets an object emit a coloured light, this defaults to a black light that has no effect on the appearance of the model. When a user 'highlights' an object the object's 'Emissive' property is set to red as this is very visible for most objects (this effect is visible for all objects that are non-white or non-red). The advantage of this approach is that a copy of the original colour of the object does not need to be stored to reset its colour. When the user is no longer hovering over an object its 'Emissive' property is reset to default.

To 'Select' a parcel, the approaches used are:

1. Drag Parcel: When the user clicks/taps on a parcel, the parcel becomes

selected and the user can manually drag the parcel along a line that minimises overlap with other parcels. The user drags upwards to move the parcel away from the brain and downwards to move the parcel towards its original position. When the user releases the mouse the parcel stays in that place. This is achieved by checking on each frame whether the mouse's 'y' co-ordinate is higher or lower that the previous frame and moving the parcel away from or towards the brain.

Mathematically, if moving away from the brain, the position of the selected parcel is updated such that:

$$v_{newposition} = v_{oldposition} + \alpha * l_{line} \tag{1}$$

$$l_{line} = (v_{originalposition} - p_{centrepoint}) \tag{2}$$

else if moving towards the brain:

$$v_{newposition} = v_{oldposition} - \alpha * l_{line} \tag{3}$$

$$l_{line} = (v_{originalposition} - p_{centrepoint}) \tag{4}$$

Where α is a heuristically defined constant. Sensible limits are given to each parcel so that users cannot drag the parcel too far or inwards past its original position.

2. Pop-out Parcel: When the user clicks/taps on a parcel, the parcel moves a fixed length along a line outwards of the brain, clicking/tapping again will either raise the parcel another length along the line or move the parcel back to its original position (this could be used to indicate groups of parcels at a time). Mathematically the selected parcel updates its

position as per:

$$v_{newposition} = v_{originalposition} + \alpha * l_{line} \tag{5}$$

$$l_{line} = (v_{originalposition} - p_{centerpoint}) \tag{6}$$

Where α is a heuristically defined constant. Clicking a parcel a second time returns it to its start position:

$$v_{newposition} = v_{originalposition} \tag{7}$$

These movements of the parcel will be animated to move along the line to its new position rather than 'teleporting' there.

Users are able to toggle between these two strategies.

3.4.3 Adding information

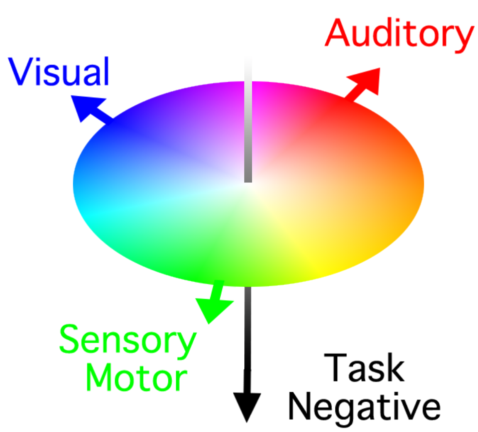


Figure 21: Colour key for Glasser et al's parcellation

In Glasser at al's parcellation, a parcel's colour is defined by a scheme that reflects broad classes of brain function - these are 'Auditory', 'Visual' and 'Sensory and Motor' (See Figure 21). Figure 21 is included as an image overlay in the Cortical Explorer with a transparent background.

Glasser et al provide supplementary neuro-anatomical results with extensive information on the information used to delineate to areas, as well as background information on their possible functions and where applicable lists studies in which the regions have been previously reported. This is valuable information but is presented in a dense text format. A key objective of this project is presenting this information in an interactive and easy to understand form. This is achieved by extracting the information from the supplementary material as shown in 'Data Mining' in Figure 13 and transforming it into a form suitable to load into MongoDB. For each parcel the following information was extracted:

- Parcel Index - index of parcel.
- Parcel Name - short name of each parcel.
- AKA (Also known as) - names this parcel has previously been referred to as.
- Parcel Description - short description of each parcel.
- New Area? - indication of whether this region has been reported previously.
- Relation Studies - papers that detailed this parcel before.

- Sections - list of 'sections' that the parcel was analysed as part of.

When a user selects a parcel, the database is queried for the parcel's information and it is displayed in a table on the screen as shown in Figure 20. Selecting a new parcel updates the information in the table to that of the selected parcel. An alternative to using a table is to display this information within the 3D scene but this is not easy to read and negatively impacts performance.

Although beneficial, the 'Related Studies' information is inconvenient to use as the names given are ambiguous and not always clear which scientific study they are referring to. Using the supplementary material to find the web-page for each paper, a link to each 'Related Study' was added to make each related study more accessible.

The functional connectivity of each area across 100 HCP subjects is also provided by Glasser et al. This data is used to extract the average strengths of functional connectivity for every pair of areas. This data is then be used to find which other areas are most strongly connected to the selected area. This list is added to the table of information for each parcel.

'Sections' and 'Connectivity' both represent collections of parcels. The parcels in each list should be highlighted within the 3D scene when the user selects a 'Section' or to show the top N 'Strongest connections. The user can toggle between two different methods of displaying these collections of parcels:

1. Pop-out parcels: As described in Section 3.4.2, each parcel 'pops out' to an elevated position above the brain. The positions can be reset by clicking the 'reset' button.

2. Re-colour parcels: All parcels that are not in the selected 'collection' are faded from view by updating their 'Emissive' property to a white colour. The colours can be reset by closing the information box for that parcel.

3.4.4 Exploding Parcels

Many parcels are difficult to select due to concave areas of the brain's surface, a key objective of this project is to offer the user an 'exploded view' on the parcels, as described in Section 2.4.1. Research into explosion diagrams indicated that a simplified approach for exploding the brain could be used, as the parcels are not organised into hierarchical components.

When the user clicks the 'explode' button, the parcels all move away from the centre point to give an expanded view of the brain. This allows the user to distinguish between parcels more easily, and gives the brain a level of transparency.

To achieve this, the 'explode' button triggers every parcel to 'pop-out' as previously described in Section 3.4.2. This is shown in Figure 20

3.4.5 Labelling Parcels

Without pre-existing knowledge of the parcellation, it is difficult for users to find specific parcels. To aid the user is searching for specific parcels, there is a labels toggle that turns labels on/off for the parcels.

The two main approaches to labelling objects within a 3D scene are:

1. On-Surface Labels: Labels appear to be written onto the surface directly, this is advantageous when the some part of the surface for each

section is clearly visible, however this is a very difficult problem to solve.

2. Labels on a pole: Labels appear at the end of a line attached to the relevant object, this makes labels more clearly visible and offers more flexibility in where the label and pole are placed. An issue with this approach is that labels themselves may occlude other parcels and/or other labels - fortunately there are ways to overcome this.

The Cortical Explorer uses the second approach. When the user clicks on the 'Label' toggle, this triggers a call to the API to retrieve the names of every parcel. For each parcel, a pole start point is calculated first by calculate the centre of the bounding box of the parcel and then finding the closest vertex on the parcel to the centre point. This vertex position is used a the starting point for the line. The end point is calculated by:

$$p_{endpoint} = p_{startpoint} + \alpha * l_{line} \tag{8}$$

$$l_{line} = p_{startpoint} - c_{centrepoint} \tag{9}$$

Where α is a heuristically defined constant and $c_{centrepoint}$ updates to either the centre point in-between the two hemispheres when both hemispheres are visible, or to the centre point of individual hemispheres when only that hemisphere is visible. A 2D canvas displaying the name of the parcel is created for each label, this is placed at the endpoint of the pole. The text label always faces the camera.

Because there are so many parcels/labels, there is a lot of occlusion between labels when the text is to be big enough to read properly. Having lots of labels bunched together overlapping each other is not ideal. To combat this issue a smart labelling algorithm is applied to my model. This moves the labels in real time around the 3D scene so that they do not overlap with other labels.

Tatzgern et al have two approaches to this smart labelling described in detail in Section 2.4.2[113]. In brief, the first involves setting up a 'pole' for each label and moving the labels along this pole until they no longer occlude other labels. The second is to assign each label to a series of planes perpendicular to the point of view and optimise the labels position within these.

The Cortical Explorer implements a simple version of this. Each label is placed at the end of a pole extruding from the parcel, and moves along this pole in real-time to a position where it does not block the view of another label. This is achieved by calculating the screen position of each label:

$$p_{screenposition} = MVP * p_{worldposition} \tag{10}$$

where MVP is the Model View Projection matrix for the camera. Calculating the size of the label for each parcel allows calculation of whether these rectangles intersect, if a label occludes other labels it is incrementally moved along its pole until it meets satisfactory conditions.

The proposed algorithm is as follows:

```
for parcels:
    while parcel.label.isOccluding:
            moveLabelAlongPole(parcel.label);
```

3.4.6 Additional Features

Alongside the core features of the application, additional features are included to enhance cognition and improve user experience.

When the parcel is displaced from its original position a 'ghost' parcel is shown. A 'ghost' parcel is a copy of the parcel that stays in its original position, is only visible when the original parcel has been displaced. The 'ghost' parcels have a different appearance to the originals. This is useful when multiple parcels have been displaced to see where in the brain they fit, allowing users to see where the parcel they are viewing has come from.

Included in the Cortical Explorer are a series of supporting brain models that were used to generate the parcellation:

- Myelin Map
- Cortical Thickness Map
- Sulc Map
- Curvature Map
- Correlation Map

These can be accessed through a 'More' drop down box in the navigation bar of the Cortical Explorer.

The alternative surface models for the brain are useful to view the parcels at different levels of inflation. Users can dynamically change the shape of the parcels to any of the following levels of inflation:

- Flat

- Pial
- Midthickness
- Inflated
- Sphere

Section 6 discusses what each of these surfaces represent.

4 Implementation

This section describes the implementation phase of this project. Section 4.1 details the 'Converter' part of 'Data Pre-processing' as seen in Figure 13. Section 4.2 details the 'Segmenter' part of 'Data Pre-processing' in Figure 13. Section 4.3 details the implementation of the Cortical Explorer web application as shown in 'Front-End', as well as detailing 'API' and 'DB', shown in Figure 13.

4.1 CIFTI to VTK

As described in Method 3.2, MATLAB was used to achieve this conversion. The MATLAB GIFTI toolbox used does not support reading CIFTI files, so these are converted to GIFTI files before use. The Connectome Workbench's command line tool offers this functionality.

CIFTI files that store data for both hemispheres in a single files are separated into two separate GIFTI files. The exact commands are shown in Appendix A.1.1. Using these commands converts CIFTI files into per-hemisphere GIFTI files. The surface files provided are already in a GIFTI format. MATLAB is then used to convert GIFTI surface files combined with GIFTI overlay files into one VTK file using the script shown in Appendix A.1.2.

The Matlab toolbox's default functionality for saving VTK files worked fine for GIFTI files that only contained surface information, this generated the models shown in Figure 6, but the colour information was lost in the this case, this was because of the way the toolbox wrote to VTK.

$saveas(gifti, filename, vtkType)$ as shown in Appendix A.1.2 makes a call to a function $mvtk_write(gifti, filename, format)$, this function failed to transfer the colour information into VTK formats because this function only checks the GIFTI object for properties named *cdata* and *color*, As described in Section 3.2.1 the GIFTI objects loaded from do not contain a *color* property. Our data does contain a *cdata* property but this does represents colour in either parcellation data or scalar data. In the case of the parcellation data, this *cdata* is used to index the *labels.rgba* property. In the case of the scalar data, *cdata* stores a scalar value that must be transformed into an rgba value. Neither are supported by $mvtk_write(gifti, filename, format)$, the relevant section of the function can be found in Appendix A.1.3.

To solve this problem an alternate function was implemented that does handle our data:

```
saveasVTK(gifti, filename, vtkType, colourType)
```

This handles propagation of colour information for both *.label.gii* (area data) and *.func.gii* (scalar data) by changing the *colourType* input, the other inputs behave as they did previously. The important change that I made was handling the two different formats of colour information present within the *gifti* input. $saveasVTK(obj, filename, format)$ makes a call to my own function:

```
mvtk_write_color(gifti, filename, format, colourType)
```

This function addresses the issues raised:

```
%—SCALARS (and LOOKUP_TABLE)
if isfield(s,'cdata') && ~isempty(s.cdata)
  //Write point data header to vtk file
    if ~point_data_hdr
        fprintf(fid,'POINT_DATA %d\n',size(s.cdata,1));
        point_data_hdr = true;
    end
    //write scalar data metadata to vtk file
    fprintf(fid,'COLOR_SCALARS color 3\n');
    //remove zeros from cdata (these represent parts of the
    brain with no data)
    no_zeros = s.cdata(s.cdata~=0);
    //normalise no_zeros to prepare scalar data for
    transformation to colour
    norm_cdata = (s.cdata - min(no_zeros)) / (max(no_zeros) -
    min(no_zeros));
    //iterate over each vertex
    for i = 1:size(s.cdata)
        switch(gtype)
            case {'scalar'}
              //set rgb value for scalar value based on 'rainbow
    ' colour schema.
                if s.cdata(i) == 0
                    r = 1;
                    g = 1;
                    b = 1;
                else
                    a = (1-norm_cdata(i))/0.2;
                    x = floor(a);
                    y = (a-x);
```

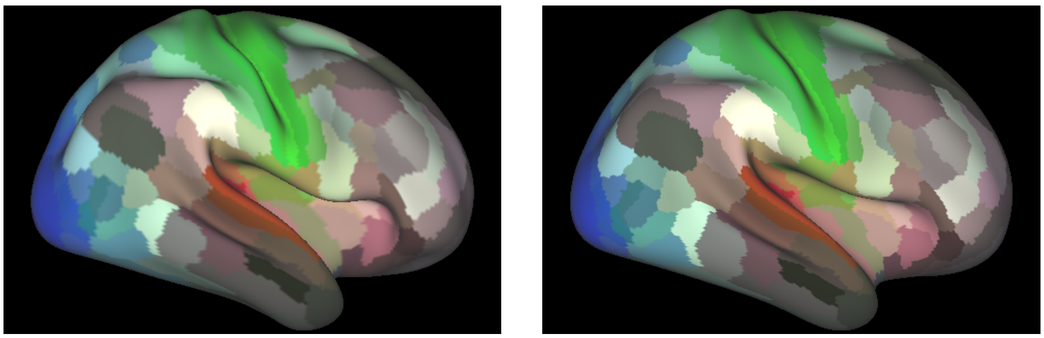
```
                    switch x
                        case 0
                            r = 1;
                            g = y;
                            b = 0;
                        case 1
                            r = 1-y;
                            g = 1;
                            b = 0;
                        case 2
                            r = 0;
                            g = 1;
                            b = y;
                        case 3
                          r = 0;
                          g = 1-y;
                          b = 1;
                        case 4
                            r = y;
                            g = 0;
                            b = 1;
                        case 5
                            r = 1;
                            g = 0;
                            b = 1;
                    end
                end
            case {'label'}
              //use cdata to index rgb values of the vertex
                j = uint32(s.cdata(i));
```

```
                if j ~= 0
                    r = s.labels.rgba(j+1, 1);
                    g = s.labels.rgba(j+1, 2);
                    b = s.labels.rgba(j+1, 3);
                else
                    r = 0;
                    g = 0;
                    b = 0;
                end
        end
        fprintf(fid, '%f %f %f\n', r, g, b);
    end
end
```

Depending on the colourType variable this function will look for either area label data (e.g. the parcellation) or scalar value data (e.g. myelin maps, cortical thickness).

The code above was originally run using 32k models of the brain (32k vertices and corresponding data points). Once the VTK files were successfully exported to VTK they were visualised in ThreeJS using the same prototype already prepared. The 'borders' of each cortical area suffered colour blending due to the low quality of the model, this blurred the edges of the parcels. In Glasser et al's visualisations they countered this problem by displaying borders between areas. Rather than convert the border files into VTK, 164k surface models for each level of brain inflation were retrieved from the HCP's tutorial data set. The parcellation data was only available in 32k, this was up-sampled to 164k using the Connectome Workbench command line utility.

The exact commands to complete this up-sampling are found in Appendix A.1.4. Using these commands a much higher quality CIFTI file was obtained, this was then converted to a VTK file. Figure 22 shows a comparison of the 32k model to the 164k model.



(a) 32k Parcellation

(b) 164k Parcellation

Figure 22: Comparison of 32k model to 164k model

4.2 Generating 360 Parcel models

The Python VTK package was used to create a novel algorithm generate 181 unique parcel meshes from the VTK mesh generated previously (Area Data VTK).

Section 3.3 outlines the steps of this algorithm. The code to complete the first step of the algorithm, reading the VTK file into a Python environment is found in Appendix A.2.1.

Parcel indexes are then iterated over to separate each parcel in turn (Steps 2 - 6 as described in 3.3). A *vtkThreshold* filter was used to separate each parcel from the rest of the surface. This filter iterates over the vertices of the object and excludes them if the threshold condition is not met. The threshold condition is determined by pointing the filter at accompanying per vertex data. Our VTK model has a colour for every vertex unique to each parcel. Instead of using a threshold on this colour information, *mvtk_write_colour*() was updated to add a set of per vertex scalar data, this would be the index of the parcel that the vertex is part of. The code added to *mvtk_write_colour* can be found in Appendix A.2.2. The *vtkThreshold* filter was then instructed to look at each vertex's parcel index and discard any vertices that are not part of the parcel (the faces that contain the discarded vertices are also discarded). The code is shown below:

```python
numParcels = 181
# Iterate over parcel indices
for i in range(0, numParcels):
    targetParcel = i
    # Extract section
        # Create vtkThreshold filter
```

```
    threshold = vtk.vtkThreshold()
        # Set threshold to only accept values for this target parcel
    threshold.ThresholdBetween(targetParcel-0.5, targetParcel+0.5)
        # Set filter to discard vertices if "parcelId" is not between thresholds
    threshold.SetInputArrayToProcess(0, 0, 0, vtk.vtkDataObject.FIELD_ASSOCIATION_POINTS, "parcelID")
        # Set data to process
    threshold.SetInputData(data)
        # Run filter
    threshold.Update()
        # Store section in variable
    section = threshold.GetOutput()
        # Convert Section to PolyData for next step
        polyFilter = vtk.vtkGeometryFilter()
        polyFilter.SetInputData(section)
        polyFilter.Update()
        section = polyFilter.GetOutput()
```

This extracted the parcel's outer surface as shown in Figure 23

A copy of the colour for the parcel is saved to reintroduce later:

```
color = list(section.GetPointData().GetScalars().GetTuple(0))
```

Step 3 as described in Section 3.3 produced erroneous results. The inner and outer surface of the parcel should not intersect as this is not anatomically correct. To solve this problem each vertex in *section* is 'warped' along its

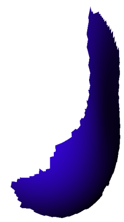


Figure 23: Result of Step 2 described in Section 3.3

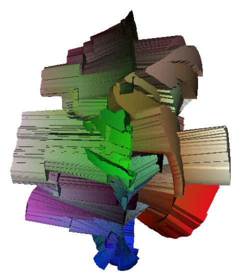


Figure 24: Results of warping the brain surface along its normal

normal vector. The scale of the 'warp' becomes important as if each vertex is warped too far then intersections between the surfaces become possible again. A $vtkWarpScalar$ filter is used to achieve this, the addition to the above code can be found in Appendix A.2.3.

The $vtkWarpScalar$ filter produced incorrect results. The filter by default searches for per vertex scalar with which to scale the warp. This resulted by default to scaling by colour information producing the result shown in Figure 24 when warping the entire brain surface for illustration.

To solve this problem a constant scalar value is added to the VTK file for each vertex, this is achieved by extending *mvtk_write_colour* with the code found in Appendix A.2.4, and updating our *vtkWarpScalar* filter to:

```
# Copy outer surface
section2 = vtk.vtkPolyData()
section2.DeepCopy(section)
# Warp Section2 along normals
warp = vtk.vtkWarpScalar()
warp.SetInputData(section2)
warp.SetScaleFactor(scaleOffset)
# *** New line directing filter to scale based on constant '
    weight'
warp.SetInputArrayToProcess(0, 0, 0, vtk.vtkDataObject.
    FIELD_ASSOCIATION_POINTS, "weight")
warp.SetUseNormal(0)
warp.Update()
section2 = warp.GetOutput()
```

This worked as expected and achieved the desired results as shown in Figure 25.

To create a complete mesh, the edges of the two sections shown in Figure 25 are stitched together. To achieve this, first the edges of each section are extracted using a *vtkFeatureEdges* filter, code for which is shown in Appendix A.2.5. This produced the results shown in Figure 26.

To create a 'strip' that joins the two edges, the lines that make up each edges are iterated over. From the corresponding line in each surface the four points that make up these two lines are accessed. From these four points, two new polygons are created that join the two lines. An overview of this

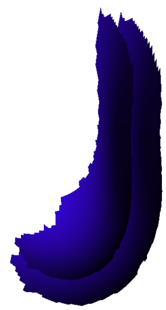


Figure 25: Inner and Outer surface of parcel

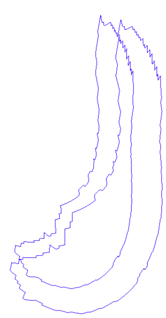


Figure 26: Extracted edges of sections

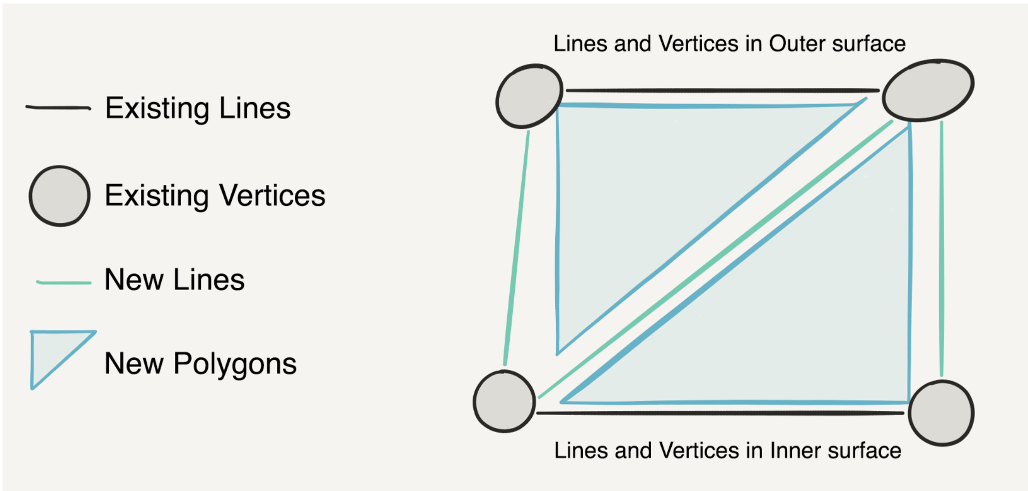


Figure 27: Overview of 'strip' generation process

can be seen in Figure 27.

The code to complete this step can be found in Appendix A.2.6. This successfully produced a strip that joins the two sections as can be seen in Figure 28.

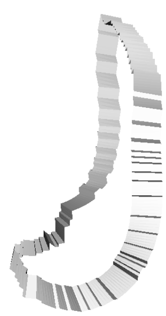


Figure 28: Strip generated

To create a complete mesh, the two sections and the strip are combined into one model, and the colour re-added to the combined mesh. This is shown in Appendix A.2.7. The mesh is then saved to a *.vtk* file using the code in Appendix A.2.8.

This algorithm successfully produces VTK models of individual parcels, the results of some of these are shown in Figure 29. This concludes the implementation of 'Data Pre-Processing' shown in Figure 13.

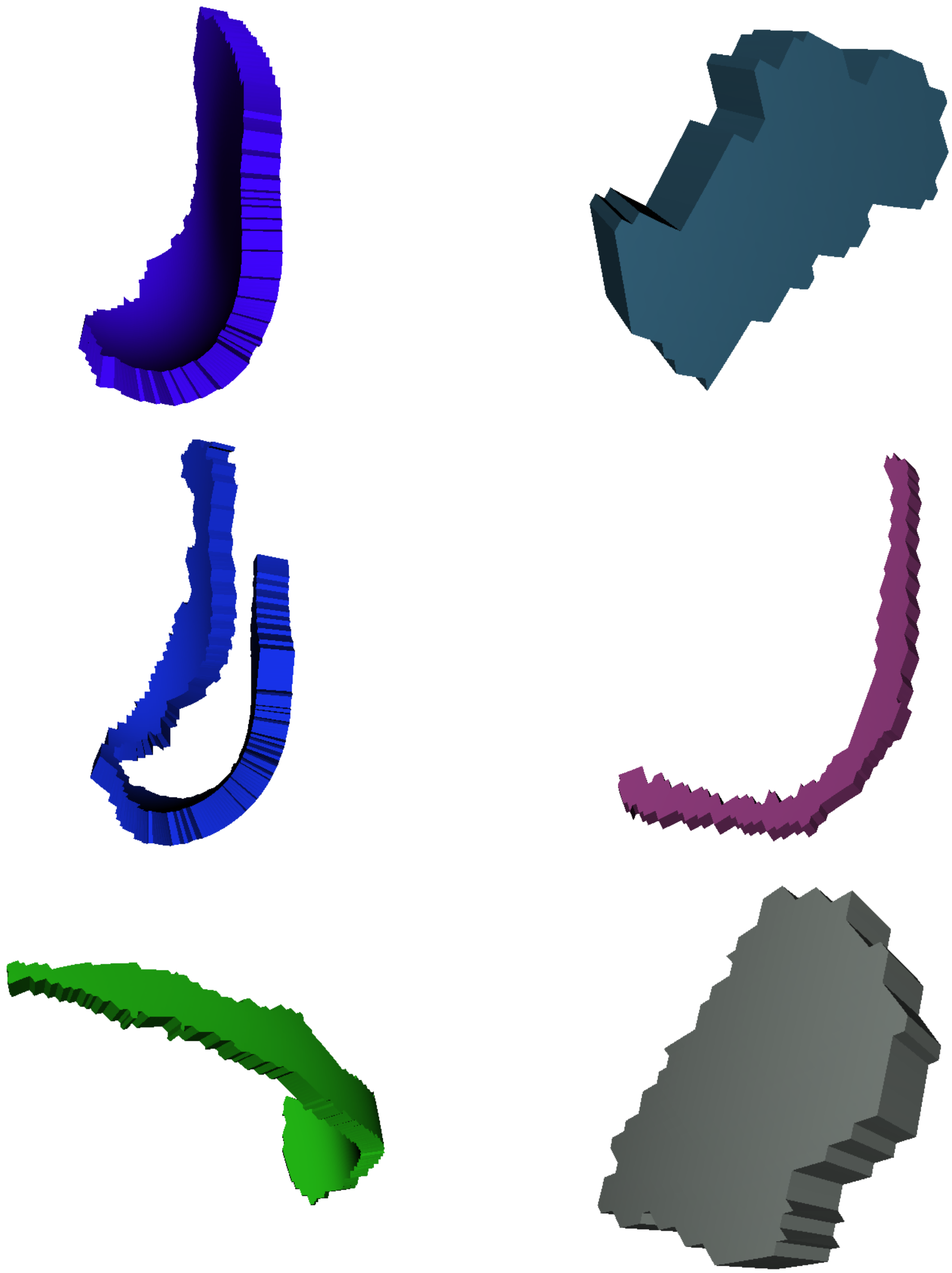


Figure 29: Some of the parcels (Generated using the 'Very Inflated' surface model)

4.3 Development of interactive, web based, 3D model

To build the 'Front-End' section from Figure 13 Javascript/HTML/CSS with Node.js and Npm were used. A simple Node.js application is set up. All Node.js projects start with a *package.json* file that defines various things such as pm modules used and scripts to run when the application starts. Webpack[42] is used to bundle application files together and serve them on an address on the local machine, this is advantageous in development as Webpack listens for changes to local code and reloads the application. *package.json*, *webpack.config.js*, the *index.html* file (entry point to webpage) and *app.js* file ('Main' in Figure 13 can be found in Appendix A.3.1).

4.3.1 ThreeJS and Basic interaction

Section 3.4.1 outlines the steps taken to visualise the parcel models in the Cortical Explorer. Appendix A.3.2 contains the code added to *app.js* that achieves what is described in Step 1 of Section 3.4.1.

Step 2 was achieved with the following addition to *app.js*:

```
//controls
controls = new THREE.OrbitControls(camera, renderer.domElement);
// Camera rotation brought to a gentle stop rather than hard
   stop
controls.enableDamping = true;
controls.dampingFactor = 0.2;
// limits on zoom distances
controls.minDistance = 100;
controls.maxDistance = 1000;
// panning on/off
```

```
controls.enablePan = true;
controls.keyPanSpeed = 7.0;
```

At this stage, The ThreeJS scene can now be controlled but there is still nothing to see in the scene. Before the parcels are loaded into the scene (Step 4), the on-screen controls were initialised. *app.js* creates a 'Ractive' object that expects to find a HTML template in *app.html* - this is where the on-screen controls are added, the controls described in Step 3 of 3.4.1 *app.html* can be found in Appendix A.3.3. The result of this is shown in Figure 30, as we can see there is a web-page with some controls available on the screen, but no parcels.

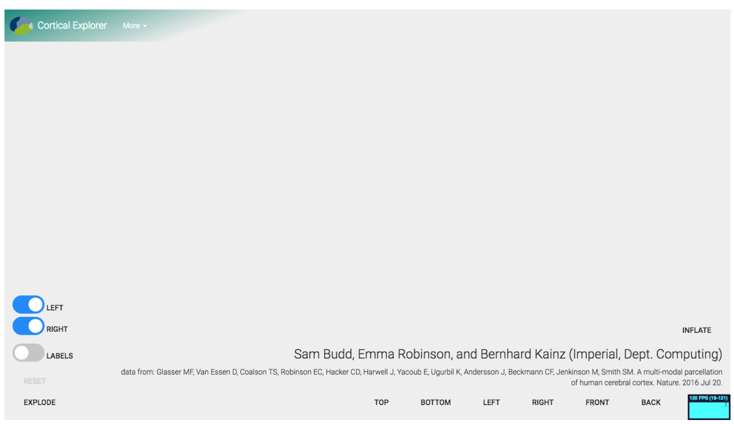


Figure 30: The Cortical Explorer after Step 3 of Section 3.4.1

For ThreeJS to work properly, two functions have to be implemented: *animate*() and *render*(). These two functions are called on every frame from in *app.js*, 'Main' in Figure 13.

```
function animate() {
```

```
    requestAnimationFrame( animate );
    controls.update();
    render();
    stats.update();
}

function render() {
    renderer.render( scene, camera );
    stats.update();
}
```

Before loading the models into the ThreeJS scene model files are loaded into the distribution by Webpack, this is done in two steps, first by 'requiring' the models in *app.js* and secondly instructing Webpack where to place the files in the distribution it serves. The code to achieve this can be found in Appendix A.3.4.

Step 4 of Section 3.4.1 can now be achieved as the 'VTKLoader' can access the model files:

```
// Create loading manager (can be used to build a loading bar)
var manager = new THREE.LoadingManager();
// initialise list of 'intersectable' objects (objects that can
   be 'highlighted' or 'selected')
var intersectable = [];
// Set number of parcels to be loaded for each hemisphere
var NUM_PARCEL = 181;
for (var i = 0; i < NUM_PARCELS; i++) {
    // load parcel i for each hemisphere into scene
    LOADER.loadLeftParcel(i, scene, manager, intersectable);
```

```
    LOADER.loadRightParcel(i, scene, manager, intersectable);
}
```

Where *loadLeftParcel* implements the steps described in Step 4 of Section 3.4.1:

```
function loadLeftParcel(parcelIndex, scene, manager, intersectable) {
    //Create generic Material for models
    let material = new THREE.MeshLambertMaterial({side: THREE.DoubleSide, vertexColors: THREE.VertexColors, shading: THREE.FlatShading});
    const loader = new THREE.VTKLoader(manager);
    loader.load( "/dist/models/parcel" + parcelIndex + "L.vtk", function ( bufferGeometry ) {
            //create geometry from vtk
        var geometry = new THREE.Geometry().fromBufferGeometry( bufferGeometry );
                //rotate geometry so brain faces the user
        geometry.rotateX( -Math.PI / 2 );
        geometry.computeVertexNormals();
                // Create a mesh with the geometry and material
        var mesh = new THREE.Mesh(geometry, material);
        mesh.name = "parcelL" + parcelIndex;
        if (parcelIndex == 0) {
            mesh.visible = false;
        }
        intersectable.push(mesh);
                // Add parcel mesh to scene.
        scene.add( mesh );
```

```
    });
}
```

The user can see the parcels of the brain (Figure 31) and rotate/pan/zoom around them (Figure 32).

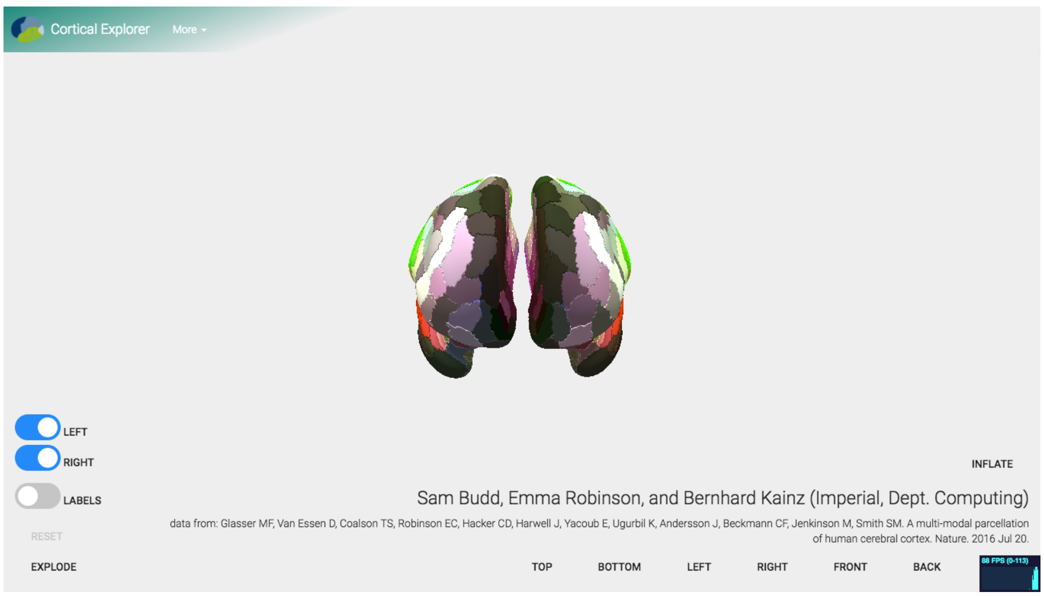
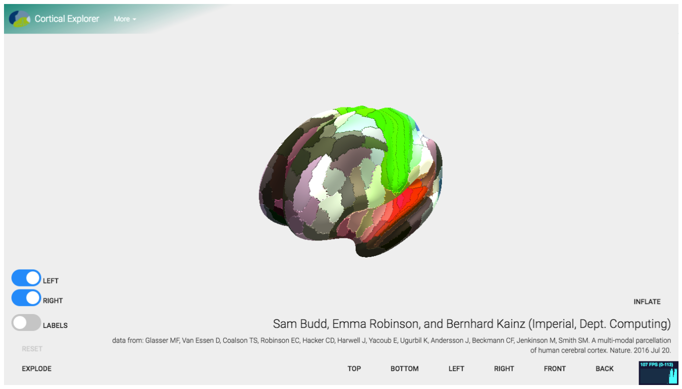
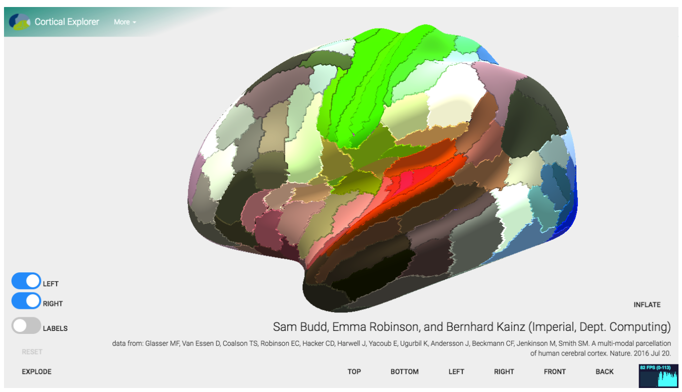


Figure 31: The Cortical Explorer with the parcels loaded

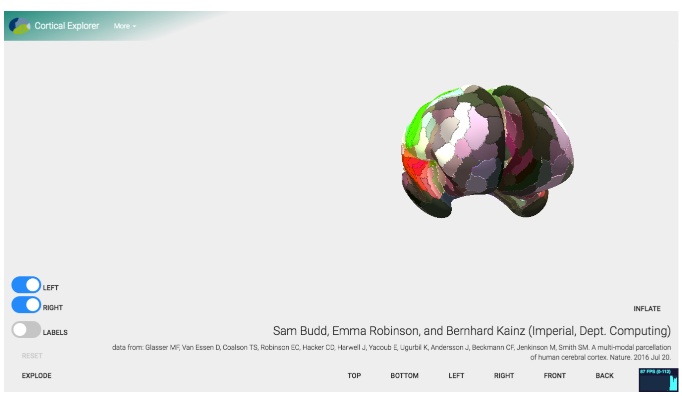
This concludes the implementation of the steps described in Section 3.4.1.



(a) Rotation



(b) Zoom



(c) Pan

Figure 32: The Cortical Explorer with basic scene interaction

4.3.2 Highlight and Select Parcels

Section 3.4.2 outlines the theory behind 'highlighting' and 'selecting' parcels. The first step of achieving this in the Cortical Explorer is by tracking the mouse position:

```
// initialise variable to store mouse position
var mouse = new THREE.Vector2(), INTERSECTED;
// intialise ThreeJS raycaster
var raycaster = new THREE.Raycaster();
// instruct onDocumentMouseMove to be called every time the user moves their mouse ('mousemove' event fired)
document.addEventListener( 'mousemove', onDocumentMouseMove, false );

//called every time the user moves their mouse
function onDocumentMouseMove( event ) {
    event.preventDefault();
    mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
    mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
}
```

On every frame, the object underneath the mouse is retrieved by adding the following lines to our *render*() function:

```
//Update ray caster to use current mouse position
raycaster.setFromCamera( mouse, camera );
// Calculate which objects are underneath the mouse
let intersects = raycaster.intersectObjects( intersectable );
// handle recolouring of parcels (highlight intersects, reset previously INTERSECTED if no longer under mouse)
```

```
INTERSECTED = PARCEL_CONTROLS.highlightOnHover(INTERSECTED,
    intersects);
```

$highlightOnHover(INTERSECTED, intersects)$ is responsible for updating the 'Emissive' property of the object under the mouse and the previous object to be under the mouse. If an object is being 'hovered' over it will change appearance according to Section 3.4.2. Appendix A.3.5 contains the implementation of $highlightOnHover(INTERSECTED, intersects)$. The results of this are shown in Figure 33.

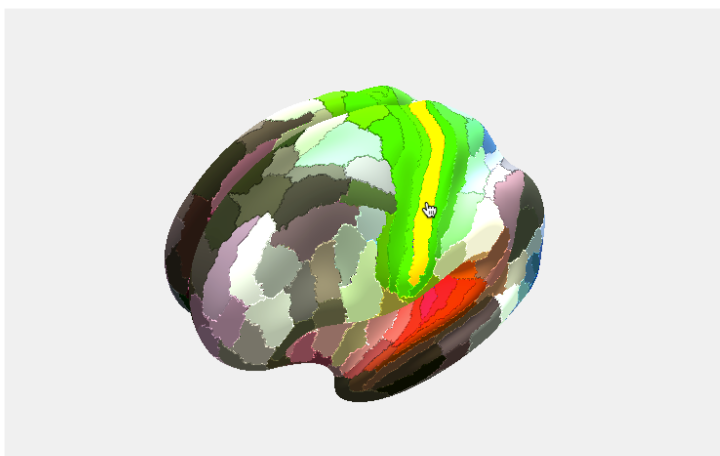


Figure 33: User 'Highlighting' an object with their mouse

As Section 3.4.2 describes, once a user is 'hovering' over a parcel, they can click/tap to 'select' the parcel.

Two methods of 'selecting' parcels are outlined in Section 3.4.2, they are implemented as follows:

1. Drag Parcel: When the user clicks/taps the screen a ray is cast into

the scene to determine the object underneath the mouse. The code for this can be found in Appendix A.3.6. When the user clicks on an object, $selectParcel(intersects, scene, controls, SELECTED)$ is called. This calculates l_{line} for the selected parcel in preparation for the user to move their mouse up or down. The implementation of $selectParcel$ can be found in Appendix A.3.7. The following lines are added to $onDocumentMouseMove$ to trigger the parcel to change position depending on whether the mouse is moving up or down:

```
if (SELECTED) {
    PARCEL_CONTROLS.dragSelected(SELECTED, mouse);
}
PARCEL_CONTROLS.PREVMOUSEY = mouse.y;
```

$dragSelected$ updates the parcel position according to the scheme defined in Section 3.4.2, the code for which is found in Appendix A.3.8. The results of this can be seen in Figure 34.

2. Pop Parcel: When the user clicks/taps on a parcel, this parcel should 'pop-out' as described in Section 3.4.2. A toggle was introduced to switch between 'selecting' methods (Drag and Pop). The following was added to $onMouseUp$ to check if:

   (a) A parcel has been selected.

   (b) The 'selecting' method is currently 'pop'.

```
if (SELECTED && parcelPopout) {
        PARCEL_CONTROLS.popoutSelected(SELECTED, scene);
```

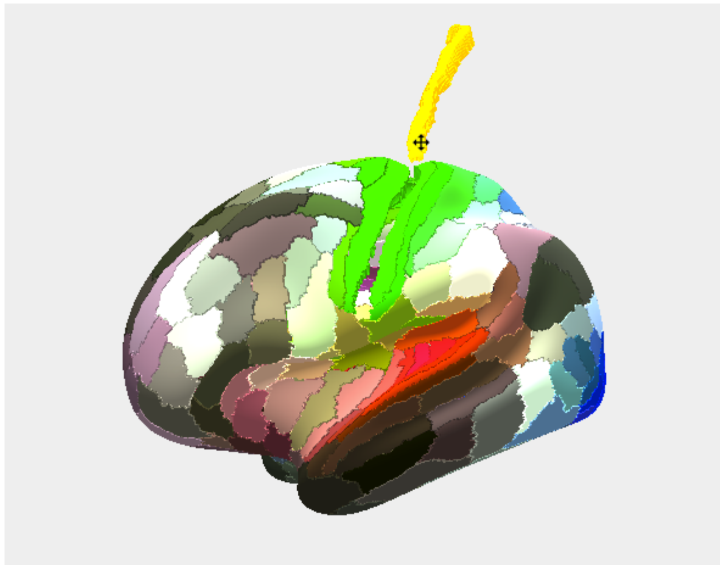


Figure 34: User dragging a parcel outwards of the brain

```
}
```

where *popoutSelected* will move the parcel along a path as described in Section 3.4.2. Using the same *extruded* variable for each parcel to check if the parcel is in its original position, the parcel moves away from the brain until it has reached *popLimit* levels away, at which point it will reset its position on the next click. $popoutSelected(SELECTED, scene)$ can be found in Appendix A.3.9. To animate the motion of the the parcel, a library call Tween[35] facilitates this by allowing creation of Tween animation objects that are updated on each call to *render*. The exact implementation of this can be found in Appendix A.3.9. The results of this are shown in Figure 35 which shows parcels 'popped-out' to two different levels.

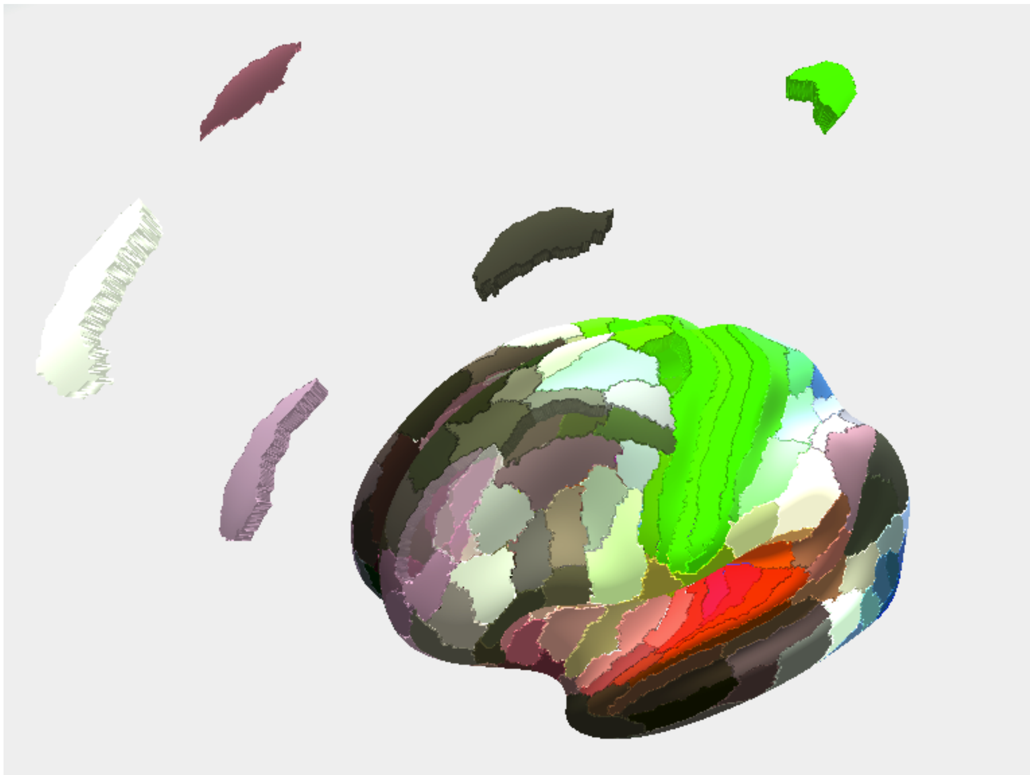


Figure 35: User has 'popped-out' parcels to different levels

This conclides the implementation of Section 3.4.2.

4.3.3 Adding Information

Section 3.4.3 describes the supporting information that the Cortical Explorer provides. The colour key shown in Figure 21 is added to the on-screen controls in *app.html*:

```
<!--color key-->
```

```
<div class= "text-right" style="z-index:20;position: absolute;
    top:60px;left:5px;margin:auto;">
     <img src="../../public/Images/ColourKeyTransparent.png"
    height="200px" width="200px" alt="Color Key">
</div>
```

The result of this addition to the Cortical Explorer is shown in Figure 36.

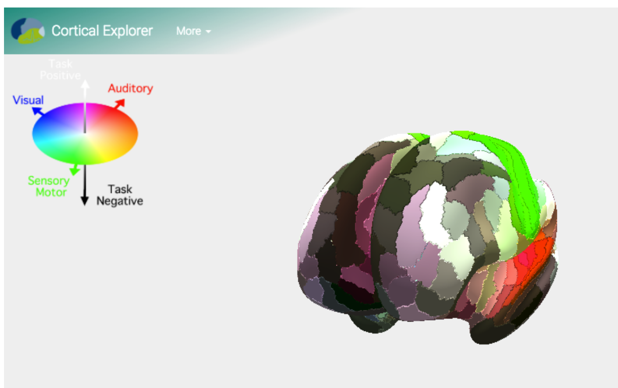


Figure 36: The Cortical Explorer with the Colour key added

As described in Section 3.4.3, the per-parcel meta-data was extracted from Glasser et al's supplementary material (Appendix B). This data was transformed into a *csv* (Comma Separated Values) format.

MongoDB

A MongoDB instance is set up to store the extracted data. Appendix A.3.10 explains the steps necessary to set up a MongoDB instance and load a *csv* file into the database. This loads in the data for each parcel as described in Section 3.4.3.

API

With the appropriate data loaded into the database the API ('API' in Figure 13) was implemented to allow the front-end to query the data from the database. Instructions for setting up the API are found in Appendix A.3.11.

Front End Interface

With the API running, the front-end can make queries to the database. The 'API' section in the 'Front-End' as shown in Figure 13 was implemented to send queries to the API.

When the user selects a parcel, a call is made to the back end to retrieve the information for that parcel:

```
function getInfoForParcel(parcel, callback) {
        http.get({
            hostname: this.API_ADDR,
            port: this.API_PORT,
            path: '/parcels/' + parcel,
            agent: true
        }, (res) => {
            res.on('data', (data) => {
                const jsonObject = JSON.parse(data);
                callback(jsonObject["0"]);
            });
        })
    }
```

This data is then passed on the calling function responsible for displaying the 'Information Box' on the screen. Ractive's two-way binding is used to display the retrieved information on the screen. The *Ractive* object created in *app.js* is updated as well as *app.html* as described in Appendix A.3.12 to achieve this. The result of this is shown in Figure 37.

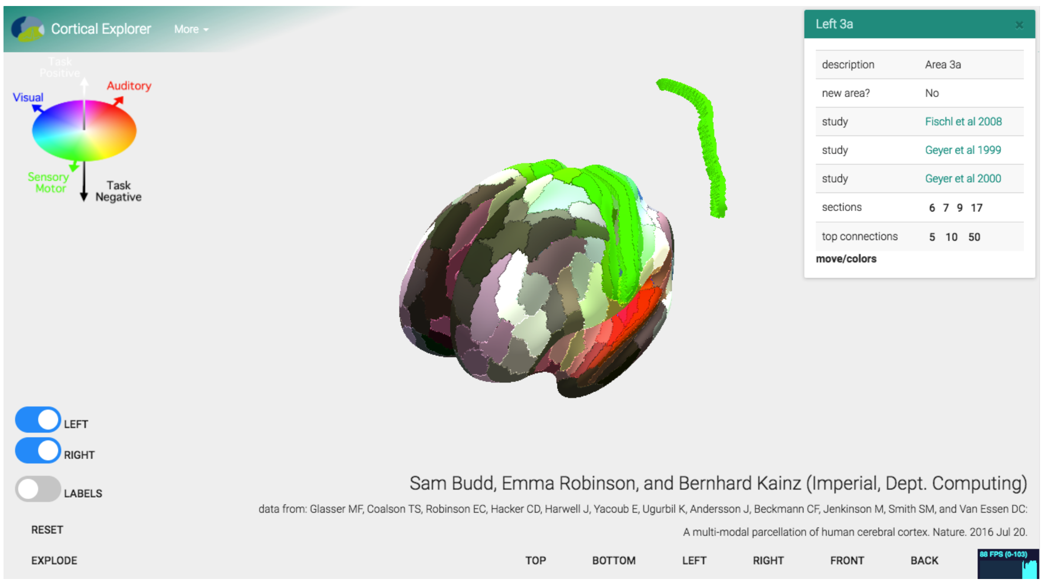


Figure 37: A Parcel is selected and it's metadata is shown

Including a link to each 'Related Study' for each parcel means users can click the 'Related Study' in the 'Information Box' to open a new tab/window leading to the PubMed[12] page for that paper.

Parcel Sections

The 'Information Box' shows which 'Sections' the parcel was present in during analyses as described in Section 3.4.3. By clicking on the 'Section' number in the 'Information Box' the other parcels that were also part of that section are indicated as described in Section 4.3.5. This is achieved by querying the API for a list of parcels also in the selected section (Appendix A.3.13) and indicating the collection of parcels as described in Section 4.3.5.

Parcel Connections

Additional data that detailed the functional connection strengths between areas was provided and used to extract a list of the strongest inter-parcel connections for each parcel. This was written to *json* and loaded this into the database. Additional 'routes' were added to the back-end API to allow querying for strongest N connections (Appendix A.3.14). Within the 'Information Box' the user has the ability to select either 5, 10 or 50 strongest connections that are indicated as described in Section 4.3.5.

This concludes the implementation of Section 3.4.3

4.3.4 Exploding Parcels

To offer an alternative view on the parcellation the user can 'explode' the brain, making many parcels easier to see and interact with as described in Section 3.4.4. Using the same approach for 'popping out' a selected parcel as detailed in Section 4.3.2, the 'explode' button triggers 'popping out' of every parcel (Appendix A.3.15). This achieved the desired 'explosion diagram' effect as can be seen in Figure 38.

This concludes the implementation of 3.4.4

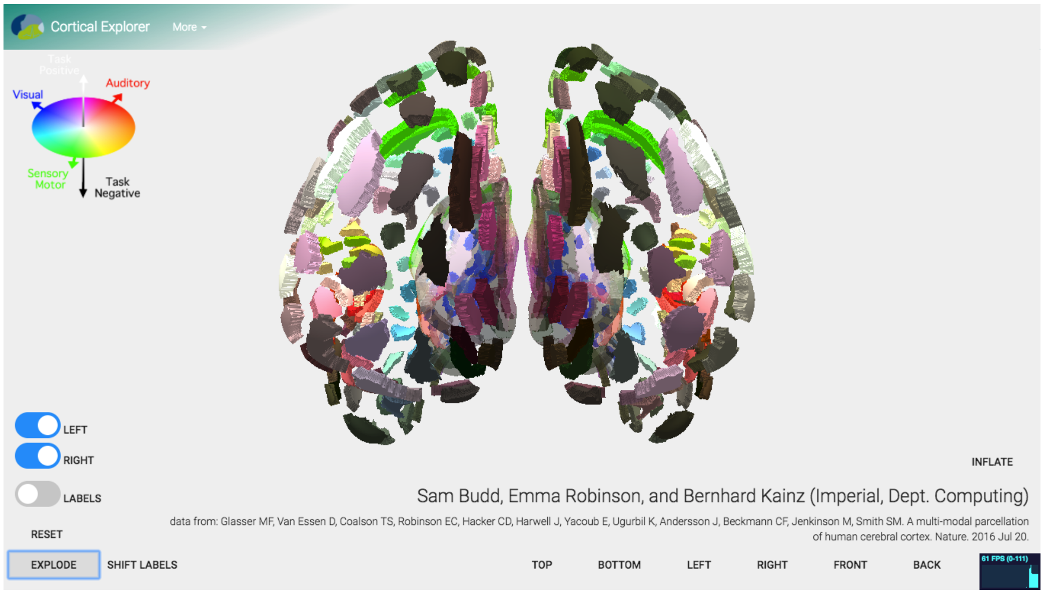


Figure 38: Explosion diagram

4.3.5 View Collections of Parcels

As previously mentioned, several features require indicating groups of parcels at a time, namely viewing 'Sections' and 'Connections'.

Exploding

'Popping-out' only the parcels in each collection makes it clearly visible which other parcels are part of the group (Appendix A.3.16). The result of this is shown in Figure 39.

Re-colouring

Updating the parcel's 'Emissive' property changes the parcels appearance. Every parcel is 'whited' out by setting the 'Emissive' property to a white

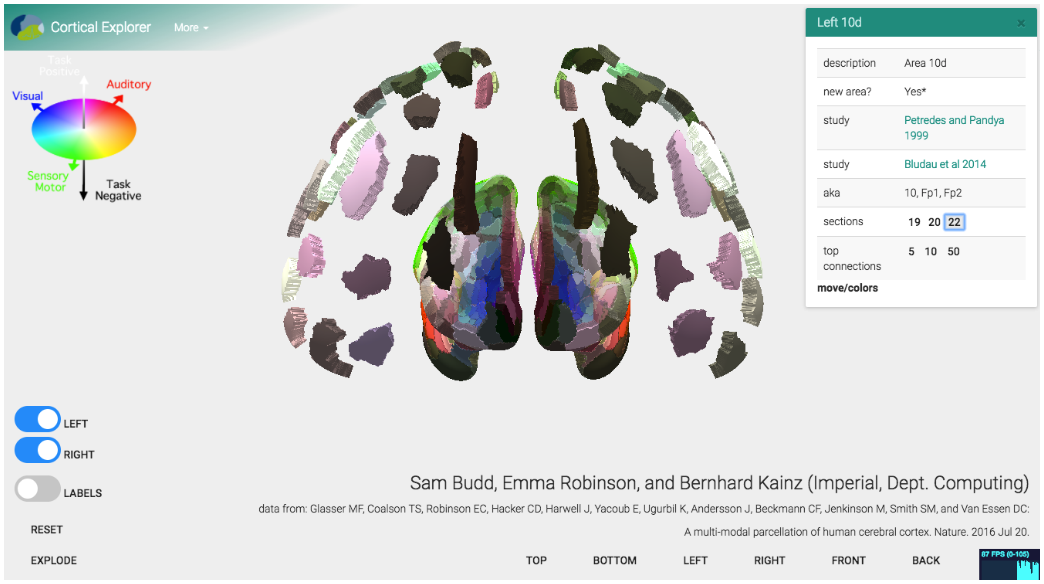


Figure 39: A 'Section' visualised by 'exploding' the parcels

colour, the parcels in the collection have their 'Emissive' property reset so they become visible over all other parcels (Appendix A.3.16). The result of this is shown in Figure 40

This concludes the implementation of viewing collections of parcels as described in Section 3.4.3.

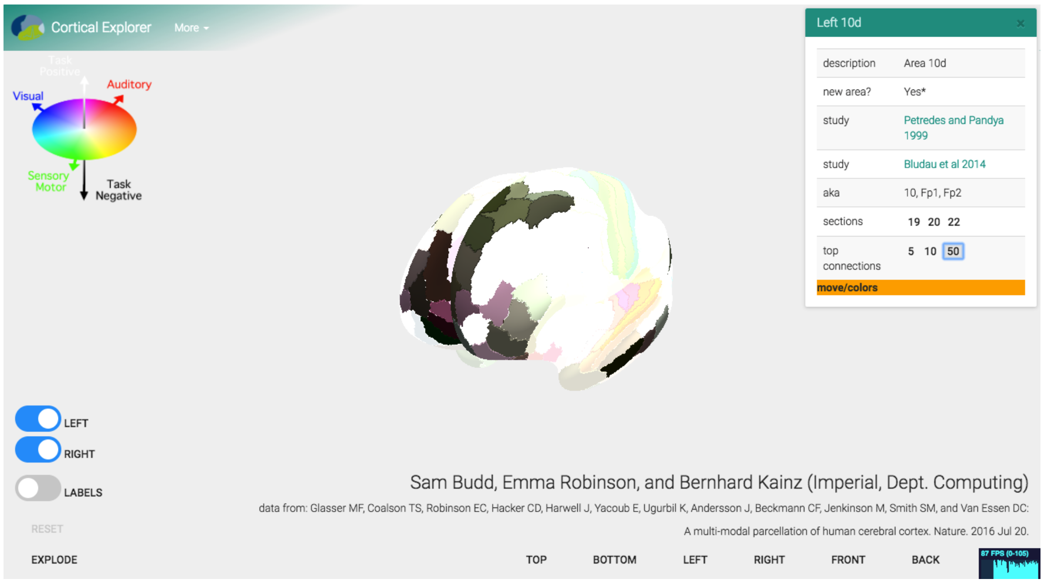


Figure 40: 50 Strongest connections visualised by 'recolouring' the parcels

4.3.6 Labelling Parcels

As described in Section 3.4.5, a big challenge in presenting this data is how best to show labels for each parcel. In the Cortical Explorer, labels are implemented following by attaching text to the end of pole that comes out of the parcel, as described in Section 3.4.5. When the user turns labels on (for the first time), the parcels in the scene are iterated over and their names are retrieved from the API. Initialising labels is detailed in Appendix A.3.17. *initLabels* uses the text for each label to add a 3D label using *addLabelFromCenter*:

```
addLabelFromCenter: function (parcel, scene, name, center) {
    //calculate start point for label pole
        var lineStart = this.getLineStartPointForParcel(parcel);
        // heuristically define constant
```

```
    var lineScale = 0.25;

    // create pole (black colour for visibility against white background)
    var material = new THREE.LineBasicMaterial({
        color: 0x000000
    });
    // create custom line geometry from line start point and calculate line end point
    var geometry = new THREE.Geometry();
    geometry.vertices.push(
        new THREE.Vector3(
            lineStart.x,
            lineStart.y,
            lineStart.z
        ),
        new THREE.Vector3(
            lineStart.x + (lineStart.x - center.x)*lineScale,
            lineStart.y + (lineStart.y - center.y)*lineScale,
            lineStart.z + (lineStart.z - center.z)*lineScale
        )
    );

    // create pole object to be added to parcel
    var pole = new THREE.Line(geometry, material);
    // instruct ThreeJS that this geometry is changed during runtime
    pole.geometry.dynamic = true;
```

```
        // create text sprite displaying parcel name
        var label = this.makeTextSprite( name, {});
        // set the position of the text to be at the end of the pole
        label.position.set(
            lineStart.x + (lineStart.x - center.x)*lineScale,
            lineStart.y + (lineStart.y - center.y)*lineScale,
            lineStart.z + (lineStart.z - center.z)*lineScale
        );
        label.name = name;
        // add text sprite to pole
        pole.add( label );
        // add pole to parcel (visible in scene on next render call)
        parcel.add( pole );
    }
```

This calculates $p_{startpoint}$ using $getLineStartPointForParcel$ (Appendix A.3.17) and $p_{endpoint}$ as described in Section 3.4.5.

$makeTextSprite(text, option)$ returns a ThreeJS 'Sprite' object. This 'Sprite' is essentially a small rectangle showing *text* that always faces the camera - The centre point of the rectangle is attached to the pole endpoint (Appendix A.3.17).

The results of this labelling are shown in Figure 41.

As can be seen in Figure 41, many of the labels become hard to read as they overlap each other. Another problem with the labels is that when a hemisphere is hidden, the labels in centre of the brain are not visible (shown

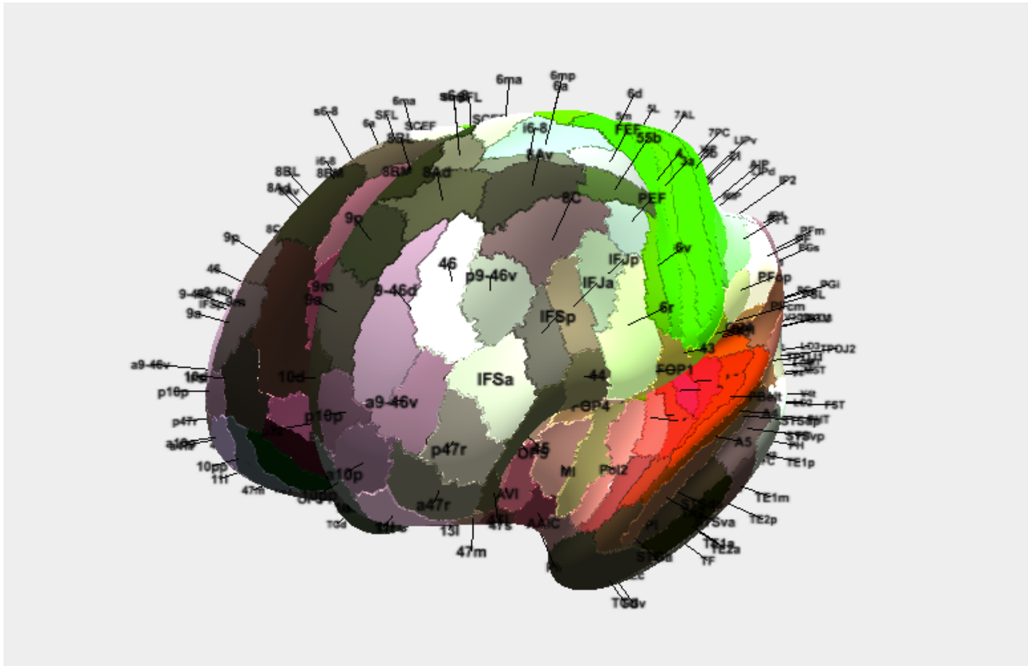


Figure 41: Labels shown for each parcel

in Figure 42).

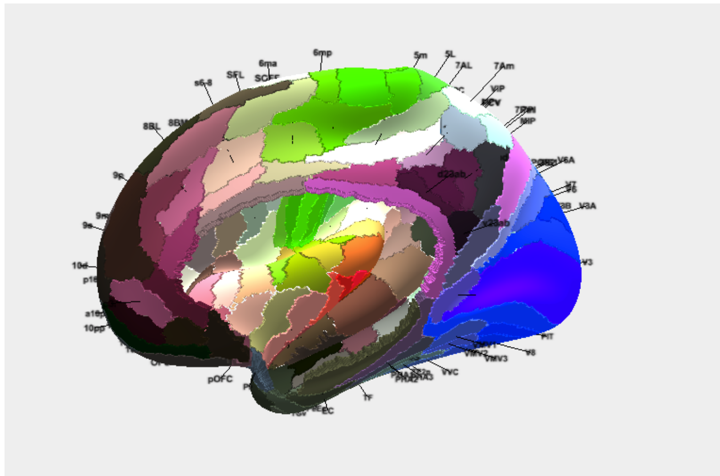


Figure 42: Labels shown while right hemisphere is hidden, some labels are hidden inside parcels

Smart Labelling

To combat the problems described above, a 'Smart Labelling' algorithm is implemented that does two things:

1. Move labels along their pole in real time to reduce occlusions - as described in Section 3.4.5.

2. Update the poles/pole direction when the user shows/hides each hemisphere of the brain to reduce occlusions between labels and parcels.

For each label the 2D screen (x, y) position of the label is calculated:

```
toScreenPosition: function(obj) {
    // create empty vector
        var vector = new THREE.Vector3();
        // get half width of viewport
```

```
        var widthHalf = 0.5*this.renderer.context.canvas.width;
        // get half height of viewport
        var heightHalf = 0.5*this.renderer.context.canvas.height
    ;
        // update object position
        obj.updateMatrixWorld();
        // copy object position into vector
        vector.setFromMatrixPosition(obj.matrixWorld);
        // project this vector using the camera to get x and y
    values between 0 and 1
        vector.project(this.camera);
        //update vector to screen coordinates
        vector.x = ( vector.x * widthHalf ) + widthHalf;
        vector.y = - ( vector.y * heightHalf ) + heightHalf;
        return {
            x: vector.x,
            y: vector.y
        };

    }
```

This extracts the 2D centre point of the label. A fixed rectangle size is chosen for all the labels. Ideally the distance from the label to the camera should be used to calculate the true size of each label rectangle (see Section 6). All pairs of rectangles is tested for overlap between them using:

```
occludes: function(label, otherLabel) {
        var labelCentre = label.screenPosition;
        var otherLabelCentre = otherLabel.screenPosition;

```

```
    var labelTopLeft = [labelCentre.x - 0.5*this.labelWidth, labelCentre.y + 0.5*this.labelHeight];
    var otherLabelTopLeft = [otherLabelCentre.x - 0.5*this.labelWidth, otherLabelCentre.y + 0.5*this.labelHeight];

    return !((labelTopLeft[0] + this.labelWidth < otherLabelTopLeft[0])
            || (otherLabelTopLeft[0] + this.labelWidth < labelTopLeft[0])
            || (labelTopLeft[1] + this.labelHeight < otherLabelTopLeft[1])
            || (otherLabelTopLeft + this.labelHeight < labelTopLeft[1]));
}
```

This gave each label a list of occluding labels. Updating the label position can be achieved in two ways.

Update Label Position in Screen space

The first option is to move the position of each label in 2D screen space until it no longer occludes other label. The new position can be back-projected into the scene to find the 3D world position to update each label to. This is possible due to the many constraints on the system. Once an non-occluding screen position for the label is found, the system of equations to be solved to retrieve the 3D label position is as follows:

$$MVP^{-1} * (x_{known}, y_{known}, z_{unknown}, 1)^T = p_{polestartpoint} + (\alpha_{unknown} * l_{poledirectionknown}) \tag{11}$$

where MVP is the Model View Projection Matrix. Solving this system of equations give us the two unknowns in our equation - the 3D world z-coordinate of the label, and the scale factor by which to scale the pole so that its endpoint is in the same place as the label.

Update Label Position in World space

The second option is to move the position of each label in 3D world space. This avoids solving the previous system of equations. However this introduces additional overheads into the algorithm. By incrementally moving each occluding parcel along its pole and then checking again whether the label is still occluding other labels, results in the labels moving to points where they don't occlude other labels.

This optimisation is triggered when the user clicks the 'Shift Labels' button:

```
optimiseLabels: function() {
      //1. get all 2D screen co-ords for each label
        this.labels.forEach((label) => {
            label.screenPosition = this.toScreenPosition(label);
            //TODO: get z-position of label using system of
    equations to properly calculate 2D rectangle size.
        });
        //2. Calculate intersections for each label
        this.labels.forEach((label) => {
```

```
            label.occludingLabelIndices = this.getOcclusionsForLabel(label);
            //TODO: factor in z co-ord to occlusions calculation due to size of label varying with distance from camera.
        });
        //Naive version: Sort labels by number of occlusions, move along pole until no more occlusions
        //3. Sort labels by number of occlusions
        this.combined = this.combined.sort(function(label, otherLabel) {
            return otherLabel.label.occludingLabelIndices.length - label.label.occludingLabelIndices.length;
        });
        //Move labels along their line until not occluding other labels.
        this.combined.forEach((object) => {
            if (object.label.occludingLabelIndices.length > this.numberOfOcclusionsAllowed) {
            this.moveLabelAlongLine(object);
            }
        });
    }
```

Where *moveAlongLine*(*object*) updates the label position until it no longer occludes other labels:

```
moveLabelAlongLine: function(object) {
        // center should change depending on brain hemispheres in view.
        var center;
```

```
        if (object.parcel.name.includes('R')) {
            center = this.CURRENT_CENTER_RIGHT;
        } else {
            center = this.CURRENT_CENTER_LEFT;
        }

        //lineStart
        var lineStart = object.parcel.geometry.vertices[object.parcel.labelStartVertexIndex];

        //scaleFactor (bit of a magic number chosen to make pole long enough to be visible outside of parcel)
        var lineScale = 0.25;

        //limit number of movements in interest of time
        var limit = 0;
        while (limit < this.loopLimit && this.labelStillOccluding(object)) {

            //calculate endpoint of new line
            var endPoint = new THREE.Vector3(
                lineStart.x + (lineStart.x - center.x)*lineScale,
                lineStart.y + (lineStart.y - center.y)*lineScale,
                lineStart.z + (lineStart.z - center.z)*lineScale
            );

            //Set position of label and length of pole
            object.label.position.set(endPoint.x, endPoint.y,
```

```
endPoint.z);
            object.pole.geometry.vertices[1] = endPoint;
            object.pole.geometry.verticesNeedUpdate = true;

            //increment line position along line
            lineScale = lineScale + this.lineIncrement;
            limit++;
        }
    }
```

Where *labelStillOccluding*(*object*) checks for occlusions:

```
labelStillOccluding: function(object) {
        //update screen position
        object.label.screenPosition = this.toScreenPosition(
    object.label);
        return (this.getOcclusionsForLabel(object.label).length
    > this.numberOfOcclusionsAllowed);
    }
```

This algorithm exposes three parameters that alter the behaviour of the labels significantly:

1. *numberOfOcclusionsAllowed*: This controls how many labels another label is allowed to occlude before it is moved along its pole (e.g. setting to zero results in all labels being updated, setting to ≥ 362 results in no labels being updated).

2. *loopLimit*: This limits how many attempts at moving the label along

its parcel are allowed before updates stop for that parcel - this is predominantly to allow the algorithm to update parcel positions in real time (discussed in Section 5.2).

3. *lineIncrement*: This controls how far along the pole each label is moved before re-checking for occlusions/loop limit reached. Increasing this results in larger motions for the labels.

By updating these parameters during run time the application can switch behaviours between the original non-smart label solution and using the smart label algorithm. This laid the foundations for building a smart labelling algorithm, there are many improvements that could be made to this discussed in Section reflabellingeval. The results of the 'Smart Labelling' are shown in Figure 43.

This implementation solves the issue illustrated in Figure 42. When hemispheres are shown/hidden, the centre point of the label poles is updated along with the pole and label positions. The result of this is shown in Figure 44

Figure 43: Smart Labels

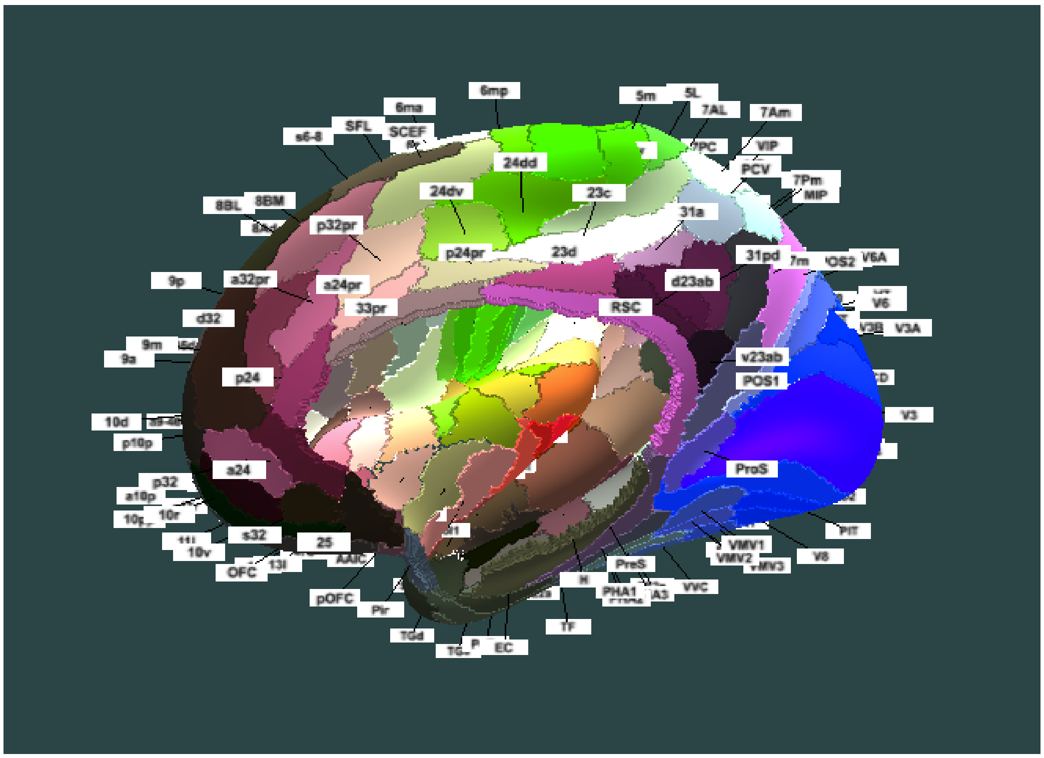


Figure 44: Smart Labels illustrating update of label when a hemisphere is hidden

4.3.7 Other Features

Ghost Parcels

When parcels are displaced for their original position a 'ghost' parcel is displayed, as described in Section 3.4.6. This is made possible by creating a copy of each parcel as its loaded, but with a different material:

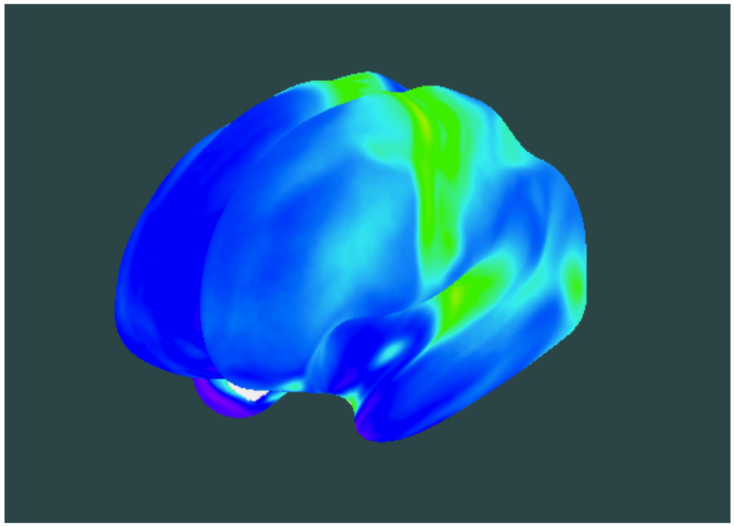
```
let ghostMaterial = new THREE.MeshLambertMaterial({ side: THREE.
    DoubleSide, vertexColors: THREE.VertexColors, shading: THREE.
    FlatShading});
ghostMaterial.transparent = true;
ghostMaterial.opacity = 0.5;

var ghost = new THREE.Mesh(geometry, ghostMaterial);
ghost.name = "ghostL" + parcelIndex;
ghost.visible = false;
scene.add(ghost);
```

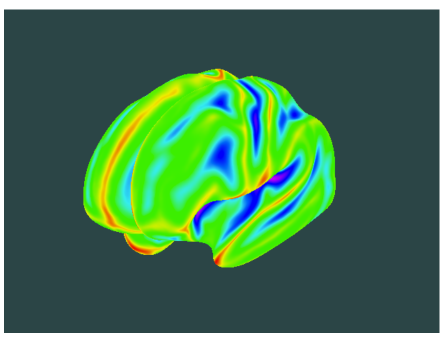
The visibility of this ghost parcel is updated when a parcel moves position. The results of this can be seen in Figure 38

Imaging Modalities

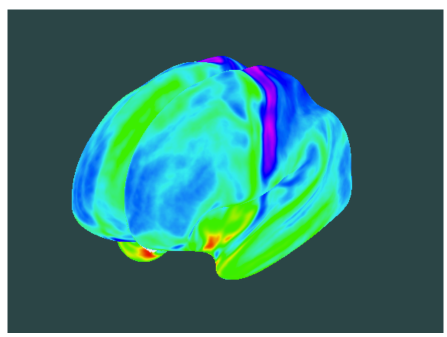
Glasser et al provided several CIFTI files containing data from different imaging modalities (described in Section 2.1.1). The navigation bar gives the users access to these. When the user selects a model from 'More' in the navigation bar, this clears the ThreeJS scene and creates a new one with the requested modality model loaded. The results of this are shown in Figure 45.



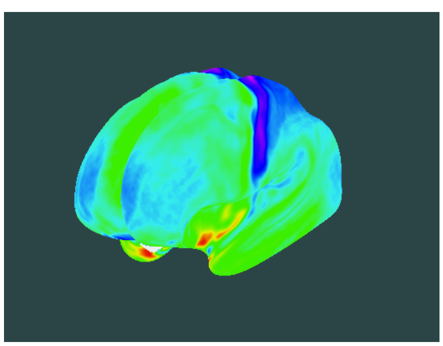
(a) Myelin



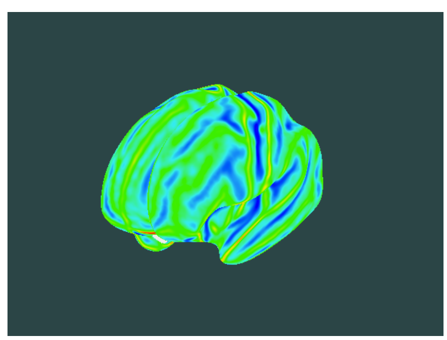
(b) Sulc



(c) Cortical Thickness



(d) Correlation



(e) Curvature

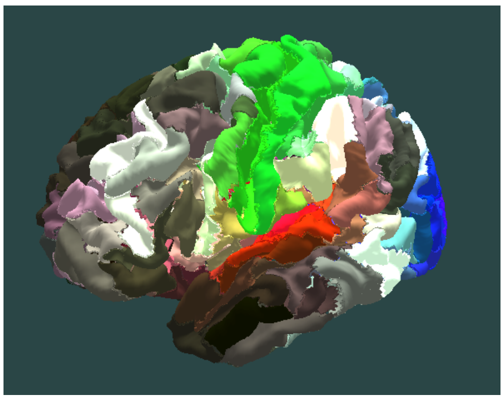
Figure 45: Various imaging modalities available to view

Alternate Surfaces

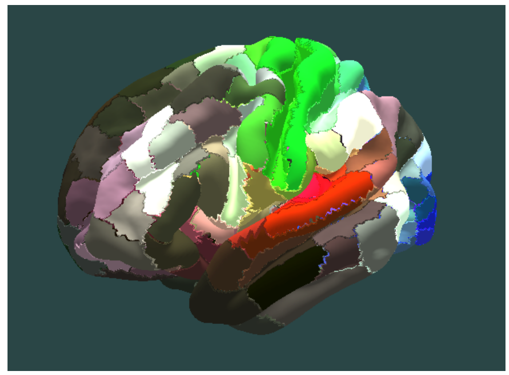
Glasser et al provide several alternate surfaces with which to visualise their data, these are described in Section 2.1.2. A button was added to 'Inflate' the parcels, this loops over 5 different surface models by loading the alternate model file when requested. The different surface models are shown in Figure 46.

4.4 Deployment

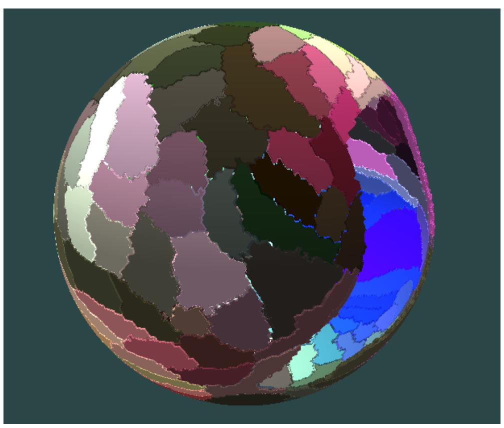
The Cortical Explorer was deployed on a virtual machine. The API and MongoDB instance were also deployed on this virtual machine. To aid in evaluation, a publicly accessible version was deployed with stable features - the survey conducted (Appendix C) was completed based on features on this instance. An 'Experimental' branch of the project was also deployed to a virtual machine accessible only within Imperial College London's network that deployed updated features as they became available.



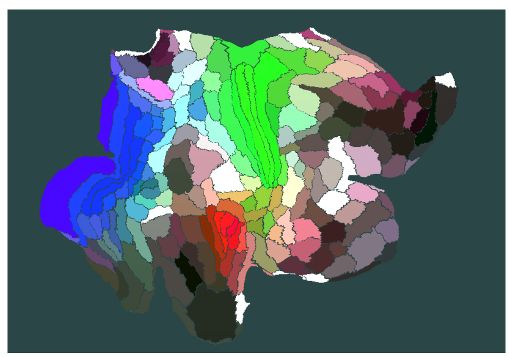
(a) Midthickness



(b) Pial



(c) Sphere



(d) Flat

Figure 46: Various surface models available for parcels

5 Evaluation and Results

This section assess the contributions of the components of the project to achieving the project objectives. Quantitative evaluation is used to assess 'Data Pre-processing' and the Cortical Explorer. A survey was created and completed by neuro-scientists from the Glasser et al team, this was done to acquire a qualitative evaluation of the Cortical Explorer and supporting tools (Appendix C). A structures interview was conducted with several members of the Glasser et al team to receive fine grained feedback on the Cortical Explorer and to discuss potential extensions of the application.

5.1 Processing Pipeline

Evaluation of 'Data Pre-Processing' component of the system (As seen in Figure 13) will consider how robust the conversion tools are, what types of files they can handle, speed, and how features of the pipeline compare to existing tools. This is a quantitative evaluation.

5.1.1 Converting from CIFTI to VTK

The 'Converter' component of 'Data Pre-processing successfully converts CIFTI metric and label files, combined with a GIFTI surface, file into a VTK file. This functionality is not available in any existing tool making it a unique solution to this problem.

The advantage this solution has over existing solutions is it's ability to convert to VTK without losing overlay information. It can handle the two main types of overlay information: *.label.gii* and *.scalar.gii*. Handling these two types of data is sufficient for this project but there are other data formats that could be handled.

This solution is an off-line solution making it unsuitable for users to upload their own custom data to be converted - Matlab is proprietary software and cannot be brought on-line for free. When collecting feedback this was a feature that Glasser et al would very much like to see. This could be achieved in two different ways:

1. CIFTI to VTK conversion brought on-line using Nibabel[24] python library instead of Matlab - This approach would likely still require the user to convert their CIFTI files to GIFTI files using the Connectome Workbench command line utility, and require intermediate storage of the CIFTI/VTK files before visualising.

2. Remove VTK from pipeline and create a ThreeJS object loader that handles CIFTI/GIFTI natively. This would allow users to upload their own data for visualisation without any intermediate representation needed - however it would also eliminate the ability to segment the surface into individual parcel volumes.

This approach is essentially a 'Proof of Concept'. This project has successfully shown it is possible to manipulate CIFTI/GIFTI files in a way that makes them suitable for web visualisation.

As a quantitative evaluation I present a table of file types converted, and the time taken to complete the conversion:

Justifying the use of the VTK format in this sub-section alone is tricky - there are many web compatible formats that would have been suitable. Only

Table 2: Supported GIFTI files for conversion on hemisphere files to VTK

Surface	Overlay	Success	Time (/s)
Inflated 32k	Metric 32k	Yes	2.03
Inflated 32k	Label 32k	Yes	3.58
Inflated 164k	Metric 164k	Yes	9.03
Inflated 164k	Label 164k	Yes	15.4

in the next section do the advantages of VTK become apparent.

5.1.2 One VTK surface to 362 VTK volumes

The algorithm described in Section 4.2 successfully splits the VTK model for each hemisphere into 181 parcel volumes. This was made possible by the flexibility of the VTK format and the features of the VTK python toolbox.

The disadvantage of this solution is that it relies on additional information being present within the VTK files (added during the conversion from GIFTI - as described in Section 4.2), this could also be included if a switch to a python-only solution was made.

As described in the previous section there is an argument to drop the intermediate VTK representation in favour of loading raw CIFTI/GIFTI files into a web-page. This would make segmenting the surface significantly harder.

An intermediate VTK allows for an more anatomically correct solution to be implemented by incorporating the cortical 'thickness' data into the model - this is described in Future work 6.

A table of the time it takes to segment the different quality of models is found in Table 3:

Table 3: Table of time taken to 'segment' model into parcels for different model qualities

Surface	Overlay	Success?	Time (/s)
Inflated 32k	Label 32k	Yes	3.93
Inflated 164k	Label 164k	Yes	13.21

5.2 Web-application

Each feature of the Cortical Explorer is assessed on how it affects frame-rates and loading-times on different devices (quantitative evaluation). The feedback gathered from the survey and structured interview is used to answer questions such as 'how useful is the Cortical explorer at communicating information' (qualitative evaluation).

The survey asked some general questions to gain an overall perspective of the quality of the application.

Figure 47 shows the responses to the Question: How useful is the Cortical Explorer to Neuroscientists? (0 = Not at all, 5 = Extremely useful). We can see that the response was positive and all users said the Cortical Explorer is useful to neuroscientists. The written answers found in Appendix C in-form us that more information should be included for the parcels and that some information such as 'Sections' is not explained well enough, Section 6 discusses what can be done to improve this.

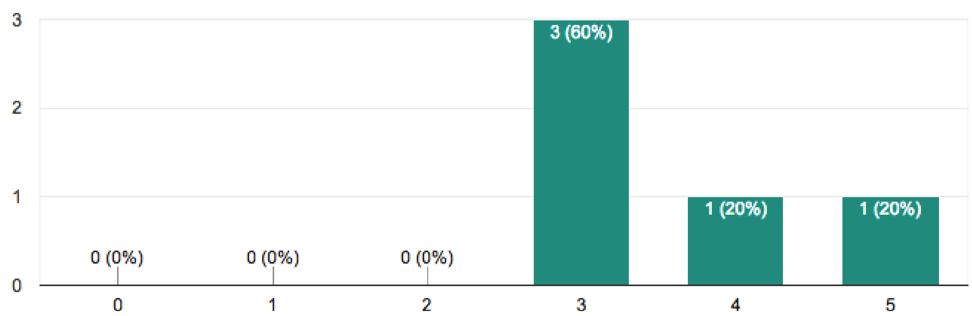


Figure 47: Responses to Question: How useful is the Cortical Explorer to Neuroscientists? 0 = Not at all, 5 = Extremely useful

Figure 48 shows the responses to the Question: How useful is the Cortical Explorer to the general public? (0 = Not at all, 5 = Extremely useful). We can see that the response was positive and all users said the Cortical Explorer is useful to the general public. Written answers suggested including more general information on the function of regions and to enrich descriptions with scientific images (e.g. auditory regions could show an illustration of the auditory system) (Appendix C).

Figure 49 shows the responses to the Question: How useful is the Cortical Explorer as a teaching tool? (0 = Not at all, 5 = Extremely useful). We can see that the response was positive and all users said the Cortical Explorer is useful as a teaching tool. Written answers to this question (Appendix C) said the Cortical Explorer "would mainly be helpful for teaching or public exploration, however, with additional features it might become very useful

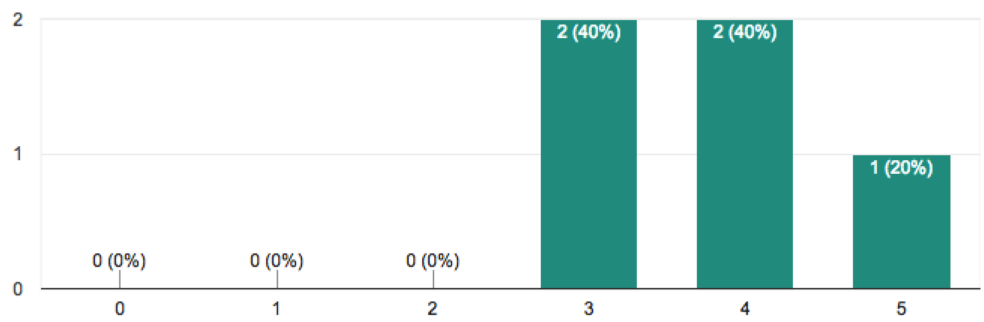


Figure 48: Responses to Question: How useful is the Cortical Explorer to the general public? 0 = Not at all, 5 = Extremely useful

for research as well".

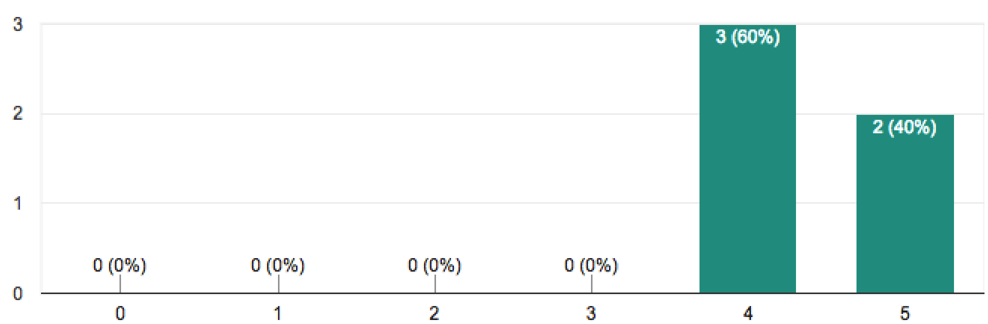


Figure 49: Responses to Question: How useful is the Cortical Explorer as a teaching tool? 0 = Not at all, 5 = Extremely useful

Basic Interaction

I had very positive feedback on the controls available, with all survey response indication that the current set of controls is sufficient to navigate the brain effectively (Appendix C). It was suggested that tutorial information for the controls be included. The on-screen controls should also become hidden once viewing models in the 'More' section when they serve no purpose. I received positive feedback on 'highlighting' parcels as it is responsive and clear. A full-screen option should also be included.

Picking Parcels

I received mixed feedback on 'picking' parcels with some responders indicating that it is one of the strongest features because of "fast response" times when selecting a parcel. However other users said this was among the weakest features as users said that "they could be simply highlighted when selected" and they 'pop-out' about "3x too far".

Information

While the information present received positive feedback overall with users saying the strongest features were "the information provided for each region (along with valid URLs to the original study findings). This is a very intuitive way of exposing users to neuro-scientific findings and extremely attractive for teaching purposes". However I also received constructive feedback such as "The information is readable, but not always clear (e.g. what does the * mean when an area is new)". Users also explained what information is best to extend the Cortical Explorer with. This is described in more detail in Section 6.

Explosion/ viewing collections

Feedback on the Explosion diagram was positive with one user including it in their 'strongest features', however one use also said that "Explode is cool, but doesn't tell you much until you click". The methods to view collections of parcels received mixed feedback with one user saying "connectivity controls were not obvious to me".

Labelling

The labels for the parcels received mixed feedback with some users saying they are "readable" but other users disagreeing saying that "The labels have a good size but appear blurry in some locations" and "Yes [they are readable/a good size] though I might customise a bit". One user agreed that the ability to move labels around manually to reduce overlap would be useful "especially for presentation purposes" while others said "It might make things messy. Rotation seems to be enough". Section 6 discusses this further.

Other features

The Cortical Explorer uses the 'inflated' surface model to display the parcels. Users gave mixed responses when asked whether viewing the other models would be useful. I implemented the ability to view the other surface models for each parcel. This had a significant negative impact on performance.

When asked when the parcels were a good thickness - users responded positively overall, users also agreed that using cortical thickness data to make the model more anatomically correct would be "nice but not essential" with one user saying "I think they are a bit too thick and using cortical thickness might be a good way to fix this."

Feedback on the brain models found in the 'More' section was constructive saying that "The introduction of the additional maps (myelin, correlation etc.) seems still immature. They need to be accompanied with colour bars and it would also be useful to have a label indicating which kind of modality is currently viewed". Extending this supporting data is discussed more in Section 6. Loading these supporting datasets had a negative impact on performance.

The survey included a Question about what type of background users would prefer, the results are summarised in Figure 50.

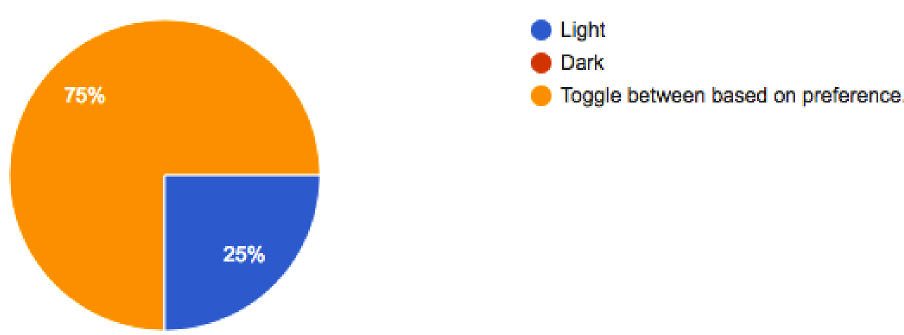


Figure 50: Responses to Question: What colour background should the Cortical Explorer have?

Table 4 details the performance of each of the core features on different devices.

Overall the Cortical Explorer performs very well and completes the project objectives. While an accomplished tool, more work can be done on the Cor-

Table 4: Feature performance

Feature	Device	FPS	Loading Time	Notes
Parcels	Laptop	85	45s, 11s from cache	FPS 70 while loading parcels
Highlighting Objects	Laptop	75	negligible	Interaction introduces drop in FPS
Selecting Objects	Laptop	65	negligible	FPS drops briefly while metadata loading
Dragging Objects	Laptop	75	negligible	FPS drops briefly while metadata loading
View Section	Laptop	75	negligible	Animation 1s
View Collection	Laptop	75	negligible	Animation 1s
Explosion	Laptop	35	negligible	Animation 1s, FPS decreases with #parcels exploded
Labels	Laptop	35	5s, negligible once loaded	
Hide Hemisphere	Laptop	110	negligible	
'More' Brains	Laptop	120	40s, 7s from cache	0 FPS and blank screen while loading
Alternate Parcel surface	Laptop	65	25s, negligible once loaded	0 FPS and blank screen while loading, memory intensive

tical Explorer and supporting tools to fulfil its potential. Some additional features are discussed in more detail in Section 6.

6 Conclusions and Future Work

The Cortical Explorer is a unique solution to visualising surface based brain data on the web. The Cortical Explorer takes advantage of the flexibility of the VTK model format to offer an new method of visualisation of Glasser et al's parcellation. The Cortical Explorer and supporting data pre-processing tools provide a foundation for a powerful brain surface visualisation tool. This could be undertaken as a PhD or research project. A suite of applications is proposed that refine existing functionality of the Cortical Explorer and extend with several new features to bring the most out the application for different audiences.

The Cortical Explorer is be targeted to two main audiences and prioritising these would affect what direction implementation prioritises:

1. The General Public: To make the Cortical Explorer of more interest to the public I propose including general neuroscience information as well as limited scientific data that is more colloquial. Including explanations of how this data was collected, the different imaging modalities and how the cortex relates the rest of the brain would be beneficial. With enough relevant information present this would make the Cortical Explorer an effective teaching tool.

2. Neuroscientists: Several features that would be useful to neuroscientists using the types of data my project has discussed were proposed to me while gathering feedback. These fall into three categories:

   (a) Analysis: providing back-end tools to perform analyses on user uploaded data.

(b) Presentation: allowing users to upload their own data for visualisation and analyses.

(c) Exploration: extending existing functionality to allow users to explore and annotate their own data.

The Cortical Explorer allows interaction with individual parcels while simultaneously allowing exploration of the whole surface, this has not yet been possible with existing visualisation tools. With the Cortical Explorer users can navigate the brain with their mouse or via touch screen interacting with parcels to view meta-data about these parcels. To aid further in exploration I propose these features to enhance the experience:

- Introduction demo: Users will have access to a Cortical Explorer Guide that explains the various features.

- Information pop-overs: Users can find more information about what each of the parcel's properties mean.

- Search: Users can search for parcels by name.

- Parcel View: Users can focus the camera on an individual parcel rather than the centre of the brain.

- View centre of the brain: At present users can show/hide each hemisphere to see the central part of the brain however an improvement on this is rotating each hemisphere away from each other to expose the central region.

The Cortical Explorer is Web-based and device-independent, this greatly increases the accessibility of Glasser et al's parcellation and supporting meta-data. Allowing users to explore per-parcel data via interaction within a structured 3D scene is an objective improvement on exploring by reading a text file. To extend the available information I propose including several other forms of per-parcel information:

- Task activity wheel: With the inclusion of task based MRI data in the parcellation this allows for the creation of a parcel 'task activity' fingerprint for each parcel. There were several tasks that each subject undertook and this data will be visualised in way similar to Figure 51

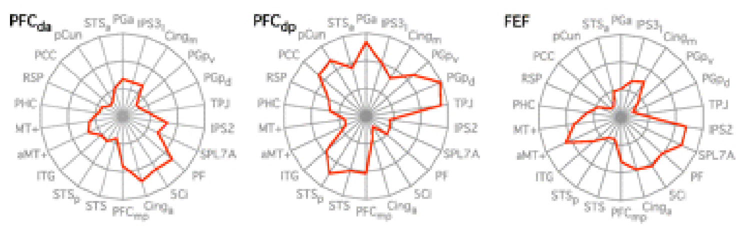


Figure 51: Example of what the 'task fingerprint' might look like [33]

- Connection strengths: At present, users can view which parcels are more strongly connected to each parcel - users will be able to see the connection strengths as well.

- Word Cloud: By mining the related studies for I can present a 'word cloud' of the most strongly associated features for that parcel.

- Cortical thickness data: Using the cortical thickness data gathered by the HCP to make our 3D parcel meshes anatomically correct.

- Tracts: visualise tract data as shown in Figure 5

The Cortical explorer works well in traditional web environments but there is scope to extend the devices it supports to enable new ways of interacting with this data:

- The Global Data Observatory: The Global Data Observatory is a collection of 64 1080p monitors arranged in stacks of four covering 320 degrees of view [18]. This unique environment allows for data visualisations to be expanded to new levels with the quantity of pixels visible. The GDO has a unique infrastructure and software structure allowing developers to leverage the system to get the most out of their application. The GDO also allows you to view a web-page spread across multiple screens and to interact with this via tablet or desktop. Bringing the Cortical Explorer into this environment could allow for more advanced features to be built that take advantage of the full suite of screens inside.

- Intel RealSense: Intel's RealSense Camera (SR300) is a custom webcam that enables control of applications via hand gestures (among other features)[17]. Integrating support for hand gestures within the Brain Explorer will add a new layer of immersion, and will make interaction with the brain models more natural. It is possible to integrate this capability in web applications (unfortunately at present the RealSense SDK is only compatible with Windows).

The 'Explosion' feature is unique to the Cortical Explorer and uses robust, well studied techniques applied in the context of brain surface visualisation.

This provides a previously unsupported way to view the brain. This could be extended by including other parts of the brain in the visualisation such as the brain stem.

Although the 'Labels' feature of the Cortical Explorer needs refinement to replicate the results of Tatzgern et al [113], the solution I present improves on existing solutions. This will be optimised to behave as closely to Tatzgern et al's solution as possible.

This project has shown that using web-based 3D visualisation frameworks that leverage WebGL is a powerful and flexible solution to building informative and interactive medical image visualisations.

This project has shown that the VTK format can be effective in facilitating new modes of presentation of medical image data.

The Cortical Explorer can be extended to allow users to upload their own data or retrieve relevant data from the BALSA database [3]. This would require the following features:

- Handling raw GIFTI/CIFTI: By implementing a ThreeJS object loaded that handles raw GIFTI/CIFTI files this would allow users to upload their own data to be visualised and explorer.

- BALSA database: By interfacing the application with the BALSA database, which hosts two types of neuro-imaging data: 'BALSA Reference' data accurately mapped to brain atlas surfaces and volumes, and 'BALSA Studies' extensively analysed neuro-imaging data associated with published figures. Users will have access to a wide range of data - this tools could become the official viewer for the BALSA database.

- Colour sliders: When viewing scalar information users should be able to change the colours and scale used to visualise the model.
- Average scalar data for parcel: Scalar data is averaged within each parcel to show per-parcel scalar results, this can be overlaid on the brain surface.
- Machine learning analysis: With more data available and users able to upload their own data, there is scope to implement some common analysis tools into the Cortical Explorer.

Overall the Cortical Explorer is a powerful tool for visualising surface based brain data that offers new ways of exploring this data. With further work the Cortical Explorer has the potential to become a widely-used tool for brain image analysis and visualisation.

References

[1] 3T How To: Structural MRI Imaging - Center for Functional MRI - UC San Diego. http://fmri.ucsd.edu/Howto/3T/structure.html.

[2] Babylon.js. http://babylonjs.com/.

[3] BALSA. https://balsa.wustl.edu/.

[4] Caret5. http://brainvis.wustl.edu/wiki/index.php/Caret:About.

[5] Connectome Workbench. http://www.humanconnectome.org/software/connectome-workbench.

[6] Explosion diagram. http://orig02.deviantart.net/f700/f/2011/058/1/7/177e6b06df8a9e3f42dc d3aklag.jpg.

[7] explosion diagram 2. https://s-media-cache-ak0.pinimg.com/originals/bc/cd/e5/bccde54736f158a5107af97cf2682c3b.png.

[8] Express - Node.js web application framework. https://expressjs.com/.

[9] Features — VTK. http://www.vtk.org/features/.

[10] FreeSurfer. https://surfer.nmr.mgh.harvard.edu/fswiki/FreeSurferAnalysisPipelineOverview

[11] Freesurfer Image. http://brainybehavior.com/neuroimaging/wp-content/uploads/2010/10/jt_itksnap_brain_aseg1.png.

[12] Home - PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed/.

[13] How Fast Is Realtime? Human Perception and Technology — PubNub. https://www.pubnub.com/blog/2015-02-09-how-fast-is-realtime-human-perception-and-technology/.

[14] How Magnetic Resonance Is Helping Doctors Diagnose Autism. https://www.mdtmag.com/news/2016/04/how-magnetic-resonance-helping-doctors-diagnose-autism.

[15] Human Connectome Project Informatics: quality control, database services, and data visualization. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3845379/.

[16] Human Connectome Project Website. http://www.humanconnectomeproject.org/.

[17] Intel RealSense. https://software.intel.com/en-us/articles/building-gesture-recognition-web-apps-with-intel-realsense-sdk/.

[18] KPMG Data Observatory — Imperial College London.

[19] Lighting and Shading. https://www.cs.uic.edu/ jbell/CourseNotes/-ComputerGraphics/LightingAndShading.html.

[20] Mango/Papaya. http://ric.uthscsa.edu/mango/index.html.

[21] MATLAB GIFTI Library. http://www.artefact.tk/software/matlab/gifti/.

[22] MongoDB for GIANT Ideas — MongoDB. https://www.mongodb.com/.

[23] Mouse Events; Raycasting with THREE.js —. https://mandemeskel.wordpress.com/2013/08/19/mouse-events-raycasting-with-three-js/.

[24] Neuroimaging in Python — NiBabel. http://nipy.org/nibabel/.

[25] Neurosynth. http://neurosynth.org/.

[26] NeuroVault Image. http://neurovault.org/media/images/457/pycortex_all/index.html.

[27] Node.js. https://nodejs.org/en/.

[28] npm. https://www.npmjs.com/.

[29] orbit controls. https://threejs.org/examples/misc_controls_orbit.html.

[30] ParaView. http://www.paraview.org/overview/.

[31] Paraview Image. http://fmri1.ee.ntu.edu.tw/lib/exe/fetch.php?cache=&media=pub:fmri:par

[32] Ractive.js. https://ractive.js.org/.

[33] Task Finger Print Image. http://jn.physiology.org/content/jn/106/3/1125/F31.medium.gif.

[34] Three.js. https://threejs.org/.

[35] TweenJS — A JavaScript library for tweening and animating HTML5 and JavaScript properties. http://www.createjs.com/tweenjs.

[36] Unity - Multiplatform - Publish your game to over 25 platforms. https://unity3d.com/unity/multiplatform.

[37] Unity3D. https://unity3d.com/.

[38] Unleashing the Power of 3D Internet — Blend4Web. https://www.blend4web.com/en/.

[39] User Stories and User Story Examples. https://www.mountaingoatsoftware.com/agile/user-stories.

[40] VTK. http://www.vtk.org/overview/.

[41] WebGL. https://www.khronos.org/registry/webgl/specs/1.0/#1.

[42] webpack. https://webpack.js.org/.

[43] What are the differences between Orthographic and Perspective views? https://blender.stackexchange.com/questions/648/what-are-the-differences-between-orthographic-and-perspective-views.

[44] What is ambient lighting? https://computergraphics.stackexchange.com/questions/375/what-is-ambient-lighting.

[45] What Is MongoDB? — MongoDB. https://www.mongodb.com/what-is-mongodb.

[46] Whitestorm.js. https://whsjs.io/#/.

[47] Workbench Commands. http://humanconnectome.org/software/workbench-command.

[48] Rouhollah O. Abdollahi, Hauke Kolster, Matthew F. Glasser, Emma C. Robinson, Timothy S. Coalson, Donna Dierker, Mark Jenkinson, David C. Van Essen, and Guy A. Orban. Correspondences between retinotopic areas and myelin maps in human visual cortex. *NeuroImage*,

99:509–524, 10 2014. http://www.ncbi.nlm.nih.gov/pubmed/24971513 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4121090 http://linkinghub.elsevier.com/retrieve/pii/S1053811914005199.

[49] C?line Amiez and Michael Petrides. Anatomical organization of the eye fields in the human and non-human primate frontal cortex. *Progress in Neurobiology*, 89(2):220–230, 10 2009. http://www.ncbi.nlm.nih.gov/pubmed/19665515 http://linkinghub.elsevier.com/retrieve/pii/S0301008209001105.

[50] K. Amunts, O. Kedo, M. Kindler, P. Pieperhoff, H. Mohlberg, N.J. Shah, U. Habel, F. Schneider, and K. Zilles. Cytoarchitectonic mapping of the human amygdala, hippocampal region and entorhinal cortex: intersubject variability and probability maps. *Anatomy and Embryology*, 210(5-6):343–352, 12 2005. http://www.ncbi.nlm.nih.gov/pubmed/16208455 http://link.springer.com/10.1007/s00429-005-0025-5.

[51] Katrin Amunts, Marianne Lenzen, Angela D Friederici, Axel Schleicher, Patricia Morosan, Nicola Palomero-Gallagher, and Karl Zilles. Broca's region: novel organizational principles and multiple receptor mapping. *PLoS biology*, 8(9), 9 2010. http://www.ncbi.nlm.nih.gov/pubmed/20877713 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2943440.

[52] Katrin Amunts, Aleksandar Malikovic, Hartmut Mohlberg, Thorsten Schormann, and Karl Zilles. Brodmann's Areas 17 and 18 Brought into Stereotaxic Space—Where and How Variable? *NeuroImage*,

11(1):66–84, 1 2000. http://www.ncbi.nlm.nih.gov/pubmed/10686118 http://linkinghub.elsevier.com/retrieve/pii/S1053811999905165.

[53] M. J. Arcaro, S. A. McMains, B. D. Singer, and S. Kastner. Retinotopic Organization of Human Ventral Visual Cortex. *Journal of Neuroscience*, 29(34):10638–10652, 8 2009. http://www.ncbi.nlm.nih.gov/pubmed/19710316 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2775458 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2807-09.2009.

[54] Deanna M. Barch, Gregory C. Burgess, Michael P. Harms, Steven E. Petersen, Bradley L. Schlaggar, Maurizio Corbetta, Matthew F. Glasser, Sandra Curtiss, Sachin Dixit, Cindy Feldt, Dan Nolan, Edward Bryant, Tucker Hartley, Owen Footer, James M. Bjork, Russ Poldrack, Steve Smith, Heidi Johansen-Berg, Abraham Z. Snyder, David C. Van Essen, and WU-Minn HCP Consortium. Function in the human connectome: Task-fMRI and individual differences in behavior. *NeuroImage*, 80:169–189, 10 2013. http://www.ncbi.nlm.nih.gov/pubmed/23684877 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4011498 http://linkinghub.elsevier.com/retrieve/pii/S1053811913005272.

[55] S. Bludau, S.B. Eickhoff, H. Mohlberg, S. Caspers, A.R. Laird, P.T. Fox, A. Schleicher, K. Zilles, and K. Amunts. Cytoarchitecture, probability maps and functions of the human frontal pole. *NeuroImage*, 93:260–275, 6 2014. http://www.ncbi.nlm.nih.gov/pubmed/23702412

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5325035 http://linkinghub.elsevier.com/retrieve/pii/S1053811913005466.

[56] K. Brodmann and Laurence. Garey. *Brodmann's 'Localisation in the cerebral cortex'*. Smith-Gordon, 1994.

[57] Julian Caspers, Karl Zilles, Simon B. Eickhoff, Axel Schleicher, Hartmut Mohlberg, and Katrin Amunts. Cytoarchitectonical analysis and probabilistic mapping of two extrastriate areas of the human posterior fusiform gyrus. *Brain Structure and Function*, 218(2):511–526, 3 2013. http://www.ncbi.nlm.nih.gov/pubmed/22488096 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3580145 http://link.springer.com/10.1007/s00429-012-0411-8.

[58] Svenja Caspers, Simon B. Eickhoff, Stefan Geyer, Filip Scheperjans, Hartmut Mohlberg, Karl Zilles, and Katrin Amunts. The human inferior parietal lobule in stereotaxic space. *Brain Structure and Function*, 212(6):481–495, 8 2008. http://www.ncbi.nlm.nih.gov/pubmed/18651173 http://link.springer.com/10.1007/s00429-008-0195-z.

[59] Svenja Caspers, Stefan Geyer, Axel Schleicher, Hartmut Mohlberg, Katrin Amunts, and Karl Zilles. The human inferior parietal cortex: Cytoarchitectonic parcellation and interindividual variability. *NeuroImage*, 33(2):430–448, 11 2006. http://www.ncbi.nlm.nih.gov/pubmed/16949304 http://linkinghub.elsevier.com/retrieve/pii/S1053811906006975.

[60] Hi-Jae Choi, Karl Zilles, Hartmut Mohlberg, Axel Schleicher, Gereon R. Fink, Este Armstrong, and Katrin Amunts. Cytoarchitectonic identification and probabilistic mapping of two distinct areas within the anterior ventral bank of the human intraparietal sulcus. *The Journal of Comparative Neurology*, 495(1):53–69, 3 2006. http://www.ncbi.nlm.nih.gov/pubmed/16432904 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3429851 http://doi.wiley.com/10.1002/cne.20849.

[61] Song-Lin Ding, Gary W. Van Hoesen, Martin D. Cassell, and Amy Poremba. Parcellation of human temporal polar cortex: A combined analysis of multiple cytoarchitectonic, chemoarchitectonic, and pathological markers. *The Journal of Comparative Neurology*, 514(6):595–623, 6 2009. http://www.ncbi.nlm.nih.gov/pubmed/19363802 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3665344 http://doi.wiley.com/10.1002/cne.22053.

[62] S. B. Eickhoff, Katrin Amunts, Hartmut Mohlberg, and Karl Zilles. The Human Parietal Operculum. II. Stereotaxic Maps and Correlation with Functional Imaging Results. *Cerebral Cortex*, 16(2):268–279, 4 2005. http://www.ncbi.nlm.nih.gov/pubmed/15888606 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhi106.

[63] S. B. Eickhoff, Axel Schleicher, Karl Zilles, and Katrin Amunts. The Human Parietal Operculum. I. Cytoarchitectonic Mapping of Subdivisions. *Cerebral Cortex*, 16(2):254–

267, 4 2005. http://www.ncbi.nlm.nih.gov/pubmed/15888607 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhi105.

[64] Alun Evans, Marco Romeo, Arash Bahrehmand, Javi Agenjo, and Josep Blat. 3D Graphics on the Web: a Survey.

[65] B. Fischl, N. Rajendran, E. Busa, J. Augustinack, O. Hinds, B. T. T. Yeo, H. Mohlberg, K. Amunts, and K. Zilles. Cortical Folding Patterns and Predicting Cytoarchitecture. *Cerebral Cortex*, 18(8):1973–1980, 8 2008. http://www.ncbi.nlm.nih.gov/pubmed/18079129 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2474454 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhm225.

[66] Marc-Oliver Gewaltig, Alexander K Kozlov, Jack L Gallant, James S Gao, Alexander G Huth, and Mark D Lescroart. Pycortex: an interactive surface visualizer for fMRI. *Frontiers in Neuroinformatics*, 9, 2015.

[67] Stefan Geyer. Materials and Methods. pages 9–13. 2004. http://link.springer.com/10.1007/978-3-642-18910-4_2.

[68] Stefan Geyer, Anders Ledberg, Axel Schleicher, Shigeo Kinomura, Thorsten Schormann, Uli B?rgel, Torkel Klingberg, Jonas Larsson, Karl Zilles, and Per E. Roland. Two different areas within the primary motor cortex of man. *Nature*, 382(6594):805–

807, 8 1996. http://www.ncbi.nlm.nih.gov/pubmed/8752272 http://www.nature.com/doifinder/10.1038/382805a0.

[69] Stefan Geyer, Axel Schleicher, and Karl Zilles. Areas 3a, 3b, and 1 of Human Primary Somatosensory Cortex. *NeuroImage*, 10(1):63–83, 7 1999. http://www.ncbi.nlm.nih.gov/pubmed/10385582 http://linkinghub.elsevier.com/retrieve/pii/S1053811999904408.

[70] Stefan Geyer, Thorsten Schormann, Hartmut Mohlberg, and Karl Zilles. Areas 3a, 3b, and 1 of Human Primary Somatosensory Cortex. *NeuroImage*, 11(6):684–696, 6 2000. http://www.ncbi.nlm.nih.gov/pubmed/10860796 http://linkinghub.elsevier.com/retrieve/pii/S1053811900905482.

[71] M. F. Glasser and D. C. Van Essen. Mapping Human Cortical Areas In Vivo Based on Myelin Content as Revealed by T1- and T2-Weighted MRI. *Journal of Neuroscience*, 31(32):11597–11616, 8 2011. http://www.ncbi.nlm.nih.gov/pubmed/21832190 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3167149 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2180-11.2011.

[72] Matthew F Glasser, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F Beckmann, Mark Jenkinson, Stephen M Smith, and David C Van Essen. A multi-modal parcellation of human cerebral cortex. *Nature Publishing Group*, 536, 2016.

[73] Matthew F. Glasser, Stamatios N. Sotiropoulos, J. Anthony Wilson, Timothy S. Coalson, Bruce Fischl, Jesper L. Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R. Polimeni, David C. Van Essen, and Mark Jenkinson. The minimal preprocessing pipelines for the Human Connectome Project. *NeuroImage*, 2013.

[74] Krzysztof J. Gorgolewski, Gael Varoquaux, Gabriel Rivera, Yannick Schwartz, Vanessa V. Sochat, Satrajit S. Ghosh, Camille Maumet, Thomas E. Nichols, Jean Baptiste Poline, Tal Yarkoni, Daniel S. Margulies, and Russell A. Poldrack. NeuroVault.org: A repository for sharing unthresholded statistical maps, parcellations, and atlases of the human brain. *NeuroImage*, 2016.

[75] Christian Grefkes, Stefan Geyer, Thorsten Schormann, Per Roland, and Karl Zilles. Human Somatosensory Area 2: Observer-Independent Cytoarchitectonic Mapping, Interindividual Variability, and Population Map. *NeuroImage*, 14(3):617–631, 9 2001. http://www.ncbi.nlm.nih.gov/pubmed/11506535 http://linkinghub.elsevier.com/retrieve/pii/S1053811901908584.

[76] Nouchine Hadjikhani, Arthur K. Liu, Anders M. Dale, Patrick Cavanagh, and Roger B.H. Tootell. Retinotopy and color sensitivity in human visual cortical area V8. *Nature Neuroscience*, 1(3):235–241, 7 1998. http://www.ncbi.nlm.nih.gov/pubmed/10195149 http://www.nature.com/doifinder/10.1038/681.

[77] D.J. Hagler, L. Riecke, and M.I. Sereno. Parietal and superior frontal visuospatial maps activated by point-

ing and saccades. *NeuroImage*, 35(4):1562–1577, 5 2007. http://www.ncbi.nlm.nih.gov/pubmed/17376706 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2752728 http://linkinghub.elsevier.com/retrieve/pii/S1053811907000109.

[78] K. A. Hansen, K. N. Kay, and J. L. Gallant. Topographic Organization in and near Human Visual Area V4. *Journal of Neuroscience*, 27(44):11896–11911, 10 2007. http://www.ncbi.nlm.nih.gov/pubmed/17978030 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2991-07.2007.

[79] John Harwell, Hester Bremen, Olivier Coulon, Donna Dierker, Richard C Reynolds, Nick Schmansky, Claudio Silva, Kevin Teich, David C Van Essen, Simon K Warfield, and Ziad S Saad. GIFTI Surface Data Format Version 1.0. 2011.

[80] Anton Henssen, Karl Zilles, Nicola Palomero-Gallagher, Axel Schleicher, Hartmut Mohlberg, Fatma Gerboga, Simon B. Eickhoff, Sebastian Bludau, and Katrin Amunts. Cytoarchitecture and probability maps of the human medial orbitofrontal cortex. *Cortex*, 75:87–112, 2 2016. http://www.ncbi.nlm.nih.gov/pubmed/26735709 http://linkinghub.elsevier.com/retrieve/pii/S0010945215003846.

[81] A HOPF. [Distribution of myeloarchitectonic marks in the frontal cerebral cortex in man]. *Journal fur Hirnforschung*, 2(4):311–33, 1956. http://www.ncbi.nlm.nih.gov/pubmed/13376888.

[82] N. Kanwisher and G. Yovel. The fusiform face area: a cortical region specialized for the perception of faces. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1476):2109–2128, 12 2006. http://www.ncbi.nlm.nih.gov/pubmed/17118927 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1857737 http://rstb.royalsocietypublishing.org/cgi/doi/10.1098/rstb.2006.1934.

[83] Bernhard Kerbl, Denis Kalkofen, Markus Steinberger, and Dieter Schmalstieg. Interactive Disassembly Planning for Complex Objects. *Computer Graphics Forum*, 34(2):287–297, 5 2015. http://doi.wiley.com/10.1111/cgf.12560.

[84] H. Kolster, R. Peeters, and G. A. Orban. The Retinotopic Organization of the Human Middle Temporal Area MT/V5 and Its Cortical Neighbors. *Journal of Neuroscience*, 30(29):9801–9820, 7 2010. http://www.ncbi.nlm.nih.gov/pubmed/20660263 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2069-10.2010.

[85] Sanjeev J. Koppal. Lambertian Reflectance. In *Computer Vision*, pages 441–443. Springer US, Boston, MA, 2014. http://link.springer.com/10.1007/978-0-387-31439-6_534.

[86] Milenko Kujovic, Karl Zilles, Aleksandar Malikovic, Axel Schleicher, Hartmut Mohlberg, Claudia Rottschy, Simon B. Eickhoff, and Katrin Amunts. Cytoarchitectonic mapping of the human dorsal extrastriate cortex. *Brain Structure and Function*, 218(1):157–172, 1 2013. http://www.ncbi.nlm.nih.gov/pubmed/22354469

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3535362 http://link.springer.com/10.1007/s00429-012-0390-9.

[87] F. Kurth, S. B. Eickhoff, A. Schleicher, L. Hoemke, K. Zilles, and K. Amunts. Cytoarchitecture and Probabilistic Maps of the Human Posterior Insular Cortex. *Cerebral Cortex*, 20(6):1448–1461, 6 2010. http://www.ncbi.nlm.nih.gov/pubmed/19822572 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2871375 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhp208.

[88] J. Larsson and D. J. Heeger. Two Retinotopic Visual Areas in Human Lateral Occipital Cortex. *Journal of Neuroscience*, 26(51):13128–13142, 12 2006. http://www.ncbi.nlm.nih.gov/pubmed/17182764 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1904390 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.1657-06.2006.

[89] A. Malikovic, K. Amunts, A. Schleicher, H. Mohlberg, S. B. Eickhoff, M. Wilms, N. Palomero-Gallagher, E. Armstrong, and K. Zilles. Cytoarchitectonic Analysis of the Human Extrastriate Cortex in the Region of V5/MT+: A Probabilistic, Stereotaxic Map of Area hOc5. *Cerebral Cortex*, 17(3):562–574, 3 2006. http://www.ncbi.nlm.nih.gov/pubmed/16603710 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhj181.

[90] Aleksandar Malikovic, Katrin Amunts, Axel Schleicher, Hartmut Mohlberg, Milenko Kujovic, Nicola Palomero-Gallagher, Simon B. Eickhoff, and Karl Zilles. Cytoarchitecture of the human lateral occipital cortex: mapping of two extrastriate areas hOc4la and hOc4lp. *Brain Structure and Function*, 221(4):1877–1897, 5 2016. http://www.ncbi.nlm.nih.gov/pubmed/25687261 http://link.springer.com/10.1007/s00429-015-1009-8.

[91] Michelle Moerel, Federico De Martino, and Elia Formisano. An anatomical and functional topography of human auditory cortical areas. *Frontiers in Neuroscience*, 8:225, 7 2014. http://www.ncbi.nlm.nih.gov/pubmed/25120426 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4114190 http://journal.frontiersin.org/article/10.3389/fnins.2014.00225/abstract.

[92] A. Morel, M.N. Gallay, A. Baechler, M. Wyss, and D.S. Gallay. The human insula: Architectonic organization and postmortem MRI registration. *Neuroscience*, 236:117–135, 4 2013. http://www.ncbi.nlm.nih.gov/pubmed/23340245 http://linkinghub.elsevier.com/retrieve/pii/S0306452213000419.

[93] P. Morosan, J. Rademacher, A. Schleicher, K. Amunts, T. Schormann, and K. Zilles. Human Primary Auditory Cortex: Cytoarchitectonic Subdivisions and Mapping into a Spatial Reference System. *NeuroImage*, 13(4):684–701, 4 2001. http://www.ncbi.nlm.nih.gov/pubmed/11305897 http://linkinghub.elsevier.com/retrieve/pii/S1053811900907158.

[94] Henrik R Nagel. Scientific Visualization versus Information Visualization.

[95] Rudolf Nieuwenhuys, Cees A. J. Broere, and Leonardo Cerliani. A new myeloarchitectonic map of the human neocortex based on data from the Vogt–Vogt school. *Brain Structure and Function*, 220(5):2551–2573, 9 2015. http://www.ncbi.nlm.nih.gov/pubmed/24924165 http://link.springer.com/10.1007/s00429-014-0806-9.

[96] Dost Öngür, Amon T. Ferry, and Joseph L. Price. Architectonic subdivision of the human orbital and medial prefrontal cortex. *The Journal of Comparative Neurology*, 460(3):425–449, 6 2003. http://www.ncbi.nlm.nih.gov/pubmed/12692859 http://doi.wiley.com/10.1002/cne.10609.

[97] Nicola Palomero-Gallagher, Simon B. Eickhoff, Felix Hoffstaedter, Axel Schleicher, Hartmut Mohlberg, Brent A. Vogt, Katrin Amunts, and Karl Zilles. Functional organization of human subgenual cortical areas: Relationship between architectonical segregation and connectional heterogeneity. *NeuroImage*, 115:177–190, 7 2015. http://www.ncbi.nlm.nih.gov/pubmed/25937490 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4801475 http://linkinghub.elsevier.com/retrieve/pii/S1053811915003535.

[98] Nicola Palomero-Gallagher, Brent A. Vogt, Axel Schleicher, Helen S. Mayberg, and Karl Zilles. Receptor architecture of human cingulate cortex: Evaluation of the four-region

neurobiological model. *Human Brain Mapping*, 30(8):2336–2355, 8 2009. http://www.ncbi.nlm.nih.gov/pubmed/19034899 http://doi.wiley.com/10.1002/hbm.20667.

[99] D N Pandya and F Sanides. Architectonic parcellation of the temporal operculum in rhesus monkey and its projection pattern. *Zeitschrift fur Anatomie und Entwicklungsgeschichte*, 139(2):127–61, 3 1973. http://www.ncbi.nlm.nih.gov/pubmed/4197942.

[100] M Petrides and D N Pandya. Dorsolateral prefrontal cortex: comparative cytoarchitectonic analysis in the human and the macaque brain and corticocortical connection patterns. *The European journal of neuroscience*, 11(3):1011–36, 3 1999. http://www.ncbi.nlm.nih.gov/pubmed/10103094.

[101] S. Pitzalis, C. Galletti, R.-S. Huang, F. Patria, G. Committeri, G. Galati, P. Fattori, and M. I. Sereno. Wide-Field Retinotopy Defines Human Cortical Visual Area V6. *Journal of Neuroscience*, 26(30):7962–7973, 7 2006. http://www.ncbi.nlm.nih.gov/pubmed/16870741 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.0178-06.2006.

[102] S. Pitzalis, M.I. Sereno, G. Committeri, P. Fattori, G. Galati, A. Tosoni, and C. Galletti. The human homologue of macaque area V6A. *NeuroImage*, 82:517–530, 11 2013. http://www.ncbi.nlm.nih.gov/pubmed/23770406 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3760586 http://linkinghub.elsevier.com/retrieve/pii/S1053811913006630.

[103] G Rajkowska and P S Goldman-Rakic. Cytoarchitectonic definition of prefrontal areas in the normal human cortex: I. Remapping of areas 9 and 46 using quantitative criteria. *Cerebral cortex (New York, N.Y. : 1991)*, 5(4):307–22. http://www.ncbi.nlm.nih.gov/pubmed/7580124.

[104] G Rajkowska and P S Goldman-Rakic. Cytoarchitectonic definition of prefrontal areas in the normal human cortex: II. Variability in locations of areas 9 and 46 and relationship to the Talairach Coordinate System. *Cerebral cortex (New York, N.Y. : 1991)*, 5(4):323–37. http://www.ncbi.nlm.nih.gov/pubmed/7580125.

[105] Claudia Rottschy, Simon B. Eickhoff, Axel Schleicher, Hartmurt Mohlberg, Milenko Kujovic, Karl Zilles, and Katrin Amunts. Ventral visual cortex in humans: Cytoarchitectonic mapping of two extrastriate areas. *Human Brain Mapping*, 28(10):1045–1059, 10 2007. http://www.ncbi.nlm.nih.gov/pubmed/17266106 http://doi.wiley.com/10.1002/hbm.20348.

[106] Friedrich Sanides and Helge Gr?fin Vitzthum. Zur Architektonik der menschlichen Sehrinde und den Prinzipien ihrer Entwicklung. *Deutsche Zeitschrift f?r Nervenheilkunde*, 187(7):680–707, 10 1965. http://link.springer.com/10.1007/BF00243937.

[107] F. Scheperjans, S. B. Eickhoff, L. Homke, H. Mohlberg, K. Hermann, K. Amunts, and K. Zilles. Probabilistic Maps, Morphometry, and Variability of Cytoarchitectonic Areas in the Human Superior Parietal Cortex. *Cerebral Cortex*, 18(9):2141–2157, 9 2008. http://www.ncbi.nlm.nih.gov/pubmed/18245042

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3140197 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhm241.

[108] F. Scheperjans, K. Hermann, S. B. Eickhoff, K. Amunts, A. Schleicher, and K. Zilles. Observer-Independent Cytoarchitectonic Mapping of the Human Superior Parietal Cortex. *Cerebral Cortex*, 18(4):846–867, 4 2008. http://www.ncbi.nlm.nih.gov/pubmed/17644831 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhm116.

[109] M. M. Schira, C. W. Tyler, M. Breakspear, and B. Spehar. The Foveal Confluence in Human Visual Cortex. *Journal of Neuroscience*, 29(28):9050–9058, 7 2009. http://www.ncbi.nlm.nih.gov/pubmed/19605642 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.1760-09.2009.

[110] Martin I. Sereno, Antoine Lutti, Nikolaus Weiskopf, and Frederic Dick. Mapping the Human Cortical Surface by Combining Quantitative T1 with Retinotopy†. *Cerebral Cortex*, 23(9):2261–2268, 9 2013. http://www.ncbi.nlm.nih.gov/pubmed/22826609 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3729202 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhs213.

[111] A T Smith, M W Greenlee, K D Singh, F M Kraemer, and J Hennig. The processing of first- and second-order motion in

human visual cortex assessed by functional magnetic resonance imaging (fMRI). *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(10):3816–30, 5 1998. http://www.ncbi.nlm.nih.gov/pubmed/9570811.

[112] J. D. Swisher, M. A. Halko, L. B. Merabet, S. A. McMains, and D. C. Somers. Visual Topography of Human Intraparietal Sulcus. *Journal of Neuroscience*, 27(20):5326–5337, 5 2007. http://www.ncbi.nlm.nih.gov/pubmed/17507555 http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.0991-07.2007.

[113] Markus Tatzgern, Denis Kalkofen, Raphael Grasset, and Dieter Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3D space. In *2014 IEEE Virtual Reality (VR)*, pages 27–32. IEEE, 3 2014. http://ieeexplore.ieee.org/document/6802046/.

[114] R B Tootell, N Hadjikhani, E K Hall, S Marrett, W Vanduffel, J T Vaughan, and A M Dale. The retinotopy of visual spatial attention. *Neuron*, 21(6):1409–22, 12 1998. http://www.ncbi.nlm.nih.gov/pubmed/9883733.

[115] R B Tootell, J D Mendola, N K Hadjikhani, P J Ledden, A K Liu, J B Reppas, M I Sereno, and A M Dale. Functional analysis of V3A and related areas in human visual cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 17(18):7060–78, 9 1997. http://www.ncbi.nlm.nih.gov/pubmed/9278542.

[116] Lazaros C. Triarhou. The Economo-Koskinas Atlas Revisited: Cytoarchitectonics and Functional Context. *Stereotactic and Functional Neurosurgery*, 85(5):195–203, 5 2007. http://www.ncbi.nlm.nih.gov/pubmed/17534132 http://www.karger.com/?doi=10.1159/000103258.

[117] Lazaros C. Triarhou. The Cytoarchitectonic Map of Constantin von Economo and Georg N. Koskinas. In *Microstructural Parcellation of the Human Cerebral Cortex*, pages 33–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. http://link.springer.com/10.1007/978-3-642-37824-9_2.

[118] D. Y. Tsao, S. Moeller, and W. A. Freiwald. Comparing face patch systems in macaques and humans. *Proceedings of the National Academy of Sciences*, 105(49):19514–19519, 12 2008. http://www.ncbi.nlm.nih.gov/pubmed/19033466 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2614792 http://www.pnas.org/cgi/doi/10.1073/pnas.0809662105.

[119] D. C. Van Essen, M. F. Glasser, D. L. Dierker, and J. Harwell. Cortical Parcellations of the Macaque Monkey Analyzed on Surface-Based Atlases. *Cerebral Cortex*, 22(10):2227–2240, 10 2012. http://www.ncbi.nlm.nih.gov/pubmed/22052704 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3500860 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhr290.

[120] D. C. Van Essen, M. F. Glasser, D. L. Dierker, J. Harwell, and T. Coalson. Parcellations and Hemispheric Asymmetries of Human Cerebral Cortex Analyzed on Surface-Based Atlases. *Cerebral Cortex*, 22(10):2241–2262, 10 2012. http://www.ncbi.nlm.nih.gov/pubmed/22047963 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3432236 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhr291.

[121] D. C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T. E J Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. W. Curtiss, S. Della Penna, D. Feinberg, M. F. Glasser, N. Harel, A. C. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. E. Petersen, F. Prior, B. L. Schlaggar, S. M. Smith, A. Z. Snyder, J. Xu, and E. Yacoub. The Human Connectome Project: A data acquisition perspective, 2012.

[122] B A Vogt, L J Vogt, D P Perl, and P R Hof. Cytology of human caudomedial cingulate, retrosplenial, and caudal parahippocampal cortices. *The Journal of comparative neurology*, 438(3):353–76, 9 2001. http://www.ncbi.nlm.nih.gov/pubmed/11550177.

[123] Brent A. Vogt and Leslie Vogt. Cytology of human dorsal midcingulate and supplementary motor cortices. *Journal of Chemical Neuroanatomy*, 26(4):301–309, 2003. http://www.sciencedirect.com/science/article/pii/S0891061803001042.

[124] Brent A. (Brent Alan) Vogt. *Cingulate neurobiology and disease.* Oxford University Press, 2009. https://global.oup.com/academic/product/cingulate-neurobiology-and-disease-9780198566960?cc=gb&lang=en&.

[125] V Vorobiev, P Govoni, G Rizzolatti, M Matelli, and G Luppino. Parcellation of human mesial area 6: cytoarchitectonic evidence for three separate areas. *The European journal of neuroscience*, 10(6):2199–203, 6 1998. http://www.ncbi.nlm.nih.gov/pubmed/9753106.

[126] Liang Wang, Ryan E.B. Mruczek, Michael J. Arcaro, and Sabine Kastner. Probabilistic Maps of Visual Topography in Human Cortex. *Cerebral Cortex*, 25(10):3911–3931, 10 2015. http://www.ncbi.nlm.nih.gov/pubmed/25452571 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4585523 https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhu277.

[127] Kevin S. Weiner, Golijeh Golarai, Julian Caspers, Miguel R. Chuapoco, Hartmut Mohlberg, Karl Zilles, Katrin Amunts, and Kalanit Grill-Spector. The mid-fusiform sulcus: A landmark identifying both cytoarchitectonic and functional divisions of human ventral temporal cortex. *NeuroImage*, 84:453–465, 1 2014. http://www.ncbi.nlm.nih.gov/pubmed/24021838 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3962787 http://linkinghub.elsevier.com/retrieve/pii/S1053811913009336.

Appendices

A Program Listings

A.1 Converting CIFTI to VTK

A.1.1 Convert CIFTI to GIFTI

For CIFTI's storing scalar data:

```
wb_command -cifti-separate CombinedHemisphereScalarData.dscalar.
    nii COLUMN -metric CORTEX_LEFT LeftScalarData.func.gii -
    metric CORTEX_RIGHT RightScalarData.func.gii
```

For CIFTI's storing label data:

```
wb_command -cifti-separate CombinedHemisphereLabelData.dlabel.
    nii COLUMN -label CORTEX_LEFT LeftScalarData.label.gii -label
     CORTEX_RIGHT RightScalarData.label.gii
```

A.1.2 Convert GIFTIs to VTK

```
//load matlab_GIFTI toolbox into MATLAB
addpath /.../matlab_GIfTI
//load right and left hemisphere surfaces
surfaceMidR = gifti('Models/164k/GIFTI/Surface/100307.R.
    midthickness.164k_fs_LR.surf.gii');
surfaceMidL = gifti('Models/164k/GIFTI/Surface/100307.L.
    midthickness.164k_fs_LR.surf.gii');
```

```
//load right and left hemisphere overlay maps
mapR = gifti('Models/164k/GIFTI/Areas/Colors_R.label.gii');
mapL = gifti('Models/164k/GIFTI/Areas/Colors_L.label.gii');
//copy vertices and faces from surface files to overlay map
mapR.vertices = surfaceMidR.vertices;
mapR.faces = surfaceMidR.faces;
mapL.vertices = surfaceMidL.vertices;
mapL.faces = surfaceMidL.faces;
//write to vtk
saveas(mapR, 'Models/164k/VTK/Areas/Midthickness_R.vtk', 'legacy
    -ascii');
saveas(mapL, 'Models/164k/VTK/Areas/Midthickness_L.vtk', 'legacy
    -ascii');
```

A.1.3 Colour information

```
%-SCALARS (and LOOKUP_TABLE)
if isfield(s,'cdata') && ~isempty(s.cdata)
    if ~point_data_hdr
        fprintf(fid,'POINT_DATA %d\n',size(s.cdata,1));
        point_data_hdr = true;
    end
    if ~isfield(s,'lut')
        lut_name = 'default';
    else
        lut_name = 'my_lut';
        if size(s.lut,2) == 3
            s.lut = [s.lut ones(size(s.lut,1),1)]; % alpha
        end
    end
```

```
    dataName = 'cdata';
    fprintf(fid,'SCALARS %s %s %d\n',dataName,'float',size(s.cdata,2));
    fprintf(fid,'LOOKUP_TABLE %s\n',lut_name);
    fmt = repmat('%f ',1,size(s.cdata,2)); fmt(end) = '';
    write_data(fid,[fmt '\n'],'float32',s.cdata');
    if ~strcmp(lut_name,'default')
        fprintf(fid,'LOOKUP_TABLE %s %d\n',lut_name,size(s.lut,1));
        if strcmp(format,'ASCII')
            % float values between (0,1)
            write_data(fid,'%f %f %f %f\n','float32',s.lut'); % rescale
        else
            % four unsigned char values per table entry
            write_data(fid,'','uint8',uint8(s.lut')); % rescale
        end
    end
end

%--COLOR_SCALARS
if isfield(s,'color') && ~isempty(s.color)
    if ~point_data_hdr
        fprintf(fid,'POINT_DATA %d\n',size(s.color,1));
        point_data_hdr = true;
    end
    dataName = 'color';
    fprintf(fid,'COLOR_SCALARS %s %d\n',dataName,size(s.color,2));
    if strcmp(format,'ASCII')
```

```
        % nValues float values between (0.1)
        fmt = repmat('%f ',1,size(s.color,2)); fmt(end) = '';
        write_data(fid,[fmt '\n'],'float32',s.color'); % rescale
    else
        % nValues unsigned char values per scalar value
        write_data(fid,'','uint8',uint8(s.color')); % rescale
    end
end
```

A.1.4 Up-sampling CIFTI

```
# Split 164k CIFTI into left and right GIFTIs
# https://www.humanconnectome.org/documentation/workbench-command/command-cifti-separate.html
wb_command -cifti-separate parcellations_VGD11b.164k_fs_LR.dlabel.nii COLUMN -label CORTEX_LEFT parcellations_VGD11b.164k_fs_LR.L.label.gii

# Create left or right 164k CIFTI from GIFTI
# https://www.humanconnectome.org/documentation/workbench-command/command-all-commands-help.html
wb_command -cifti-create-label L.Template.dlabel.nii -left-label parcellations_VGD11b.164k_fs_LR.L.label.gii

# Resample 32k CIFTI to 164k CIFTI
# https://www.humanconnectome.org/documentation/workbench-command/command-cifti-resample.html
wb_command -cifti-resample Q1-Q6_RelatedParcellation210.CorticalAreas_dil_Colors.32k_fs_LR.dlabel.nii COLUMN L.Template.dlabel.nii COLUMN ADAP_BARY_AREA ENCLOSING_VOXEL
```

```
Colors_L.164k.dlabel.nii -left-spheres Q1-Q6_R440.L.sphere.32
k_fs_LR.surf.giiQ1-Q6_R440.L.sphere.164k_fs_LR.surf.gii -left
-area-surfs Q1-Q6_R440.L.midthickness.32k_fs_LR.surf.gii Q1-
Q6_R440.L.midthickness.164k_fs_LR.surf.gii
```

A.2 Generating Brain Parcel Models

A.2.1 Loading VTK model

```python
# Import VTK Python package
import vtk
# Load VTK file
reader = vtk.vtkPolyDataReader()
reader.SetFileName('Models/164k/VTK/Areas/Midthickness_L.vtk')
reader.ReadAllVectorsOn()
reader.ReadAllScalarsOn()
reader.Update()
data = reader.GetOutput()
```

A.2.2 Adding parcel index as scalar data to VTK

```matlab
% ADD scalar as parcel number
if strcmpi(gtype, 'label')
    fprintf(fid,'SCALARS %s %s 1\n','parcelID','float');
    fprintf(fid,'LOOKUP_TABLE default\n');
    for i = 1:size(s.cdata)
        l = uint32(s.cdata(i));
        fprintf(fid, '%f\n', l);
    end
```

```
end
```

A.2.3 Warping section1

```
# Copy outer surface
section2 = vtk.vtkPolyData()
section2.DeepCopy(section)
# Warp Section2 along normals
warp = vtk.vtkWarpScalar()
warp.SetInputData(section2)
# Set scale factor to empirically define constant
warp.SetScaleFactor(scaleOffset)
warp.SetUseNormal(0)
warp.Update()
section2 = warp.GetOutput()
```

A.2.4 Fixing warp

```
if strcmpi(gtype, 'label')
    fprintf(fid,'SCALARS %s %s 1\n','weight','float');
    fprintf(fid,'LOOKUP_TABLE default\n');
    for i = 1:size(s.cdata)
        fprintf(fid, '%f\n', 10.0);
    end
end
```

A.2.5 Extracting the Edges

```
# Extract edges of section1 for sewing together
edges = vtk.vtkFeatureEdges()
edges.SetInputData(section)
edges.BoundaryEdgesOn()
edges.FeatureEdgesOff()
edges.ManifoldEdgesOff()
edges.NonManifoldEdgesOff()
edges.Update()
edges1 = edges.GetOutput()
# Extract edges of section2 for sewing together
edges.SetInputData(section2)
edges.Update()
edges2 = edges.GetOutput()
```

A.2.6 Generating strip

```
# Connect edges together
# create empty set of points
points = vtk.vtkPoints()
# create empty set of polygons
polygons = vtk.vtkCellArray()
# initialise index tracker
j = 0
# init traversal of lines in edge
edges1.GetLines().InitTraversal()
idList1 = vtk.vtkIdList()
edges2.GetLines().InitTraversal()
idList2 = vtk.vtkIdList()
# Iterate over lines in edge
while(edges1.GetLines().GetNextCell(idList1) and edges2.GetLines
```

```
().GetNextCell(idList2)):
    pointsInLine = []
        # Get the four points that make up these two lines
    for pointId in range(0, idList1.GetNumberOfIds()):
        pointsInLine.append(edges1.GetPoint(idList1.GetId(pointId)))
        pointsInLine.append(edges2.GetPoint(idList2.GetId(pointId)))
    p0 = pointsInLine[0]
    p1 = pointsInLine[1]
    p2 = pointsInLine[2]
    p3 = pointsInLine[3]
        # add points to strip
    points.InsertNextPoint(p0)
    points.InsertNextPoint(p1)
    points.InsertNextPoint(p2)
    points.InsertNextPoint(p3)

        # Creaate first polygon between points and add to strip
    polygon1 = vtk.vtkPolygon()
    polygon1.GetPointIds().SetNumberOfIds(3)
    polygon1.GetPointIds().SetId(0, 4*j)
    polygon1.GetPointIds().SetId(1, 4*j + 1)
    polygon1.GetPointIds().SetId(2, 4*j + 3)
    polygons.InsertNextCell(polygon1)

        # Create second polygon between points and add to strip
    polygon2 = vtk.vtkPolygon()
    polygon2.GetPointIds().SetNumberOfIds(3)
    polygon2.GetPointIds().SetId(0, 4*j)
```

```
    polygon2.GetPointIds().SetId(1, 4*j + 2)
    polygon2.GetPointIds().SetId(2, 4*j + 3)
    polygons.InsertNextCell(polygon2)
    j = j+1

# create strip from points and polygons
strip = vtk.vtkPolyData()
strip.SetPoints(points)
strip.SetPolys(polygons)
```

A.2.7 Combining

```
# 4. Combine section, warped section and edges strip
appendFilter = vtk.vtkAppendPolyData()
appendFilter.AddInputData(section)
appendFilter.AddInputData(section2)
appendFilter.AddInputData(strip)
appendFilter.Update()
parcel = appendFilter.GetOutput()
# Remove duplicate points and polygons
clean = vtk.vtkCleanPolyData()
clean.SetInputData(parcel)
clean.Update()
parcel = clean.GetOutput()
```

```
colors = vtk.vtkUnsignedCharArray()
colors.SetNumberOfComponents(3)
colors.SetName("Colors")
numPoints = parcel.GetNumberOfPoints()
```

```
for j in range(0, 3):
    color[j] = int(color[j])
color = tuple(color)
for j in range(0, numPoints):
    colors.InsertNextTupleValue(color)
parcel.GetPointData().SetScalars(colors)
```

A.2.8 Write to VTK

```
# Write to file
writer = vtk.vtkPolyDataWriter()
writer.SetInputData(parcel)
writer.SetFileName('Models/164k/VTK/'+ folderName +'/parcel' +
    str(targetParcel) + '.vtk')
writer.Write()
```

A.3 Application Development

A.3.1 App Setup

Package.json:

```
{
  "name": "cortical-front",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "webpack -w & node node_modules/node-static/bin/cli
    .js"
```

```
  },
  "author": "Samuel Budd",
  "license": "ISC",
  "devDependencies": {
    "babel-core": "^6.24.1",
    "babel-loader": "^7.0.0",
    "babel-preset-es2015": "^6.24.1",
    "bootstrap": "^3.3.7",
    "bootstrap-material-design": "^0.5.10",
    "file-loader": "^0.11.1",
    "jquery": "^3.2.1",
    "node-static": "^0.7.9",
    "raw-loader": "^0.5.1",
    "requirejs": "^2.3.3",
    "three": "^0.85.1",
    "webpack": "^2.4.1"
  },
  "dependencies": {
    "ractive": "^0.8.12"
  }
}
```

Webpack.config.js:

```
module.exports = {
    entry: {
        js: './app/js/app.js'
    },
    output: {
        filename: './dist/js/app-bundle.js'
```

```
    },
    module: {
        loaders: [
            {
                test: /\.js$/,
                exclude: /(node_modules|bower_components)/,
                loader: 'babel-loader',
                // If you're following the tutorial for the first time,
                // you will be using Babel v6 and thus you need to add this extra configuration
                query: {
                    presets: ['es2015']
                }
            },
            {
                test: /\.html$/,
                loader: 'raw-loader'
            },
            {
                test: /\.ico$/,
                loader: 'raw-loader'
            }
        ]
    }
};
```

Index.html:

```
<!DOCTYPE html>
```

```html
<html>
<head>
    <meta charset="UTF-8">
    <title>Cortical Explorer</title>
    <link rel="stylesheet" href="./node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Roboto:300,400,500,700">
    <link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/icon?family=Material+Icons">
    <link rel="stylesheet" type="text/css" href="./node_modules/bootstrap-material-design/dist/css/bootstrap-material-design.css">
    <link rel="stylesheet" type="text/css" href="./node_modules/bootstrap-material-design/dist/css/ripples.min.css">
</head>
<body>
<script src="app/js/three_utils/three.min.js"></script>
<script src="app/js/three_utils/OrbitControls.js"></script>
<script src="app/js/three_utils/VTKLoader.js"></script>
<script src="app/js/three_utils/Detector.js"></script>
<script src="app/js/three_utils/stats.min.js"></script>
<script src='app/js/three_utils/inflate.min.js'></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/tween.js/16.3.5/Tween.min.js"></script>
<script src="https://cdn.jsdelivr.net/lodash/4.17.4/lodash.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/
    bootstrap.min.js"></script>

<div id="app" style="height:0;z-index: 100000;"></div>
<script src="dist/js/app-bundle.js"></script>

</body>
</html>
```

App.js:

```
import Ractive from 'ractive';
import html from './views/app.html';

var App = new Ractive({
    el: '#app',
    template: html,
    data: {},
});

export default App;
```

A.3.2 Initialise Three.js

```
if (!Detector.webgl) Detector.addGetWebGLMessage();
// Declare Variables
var container, stats, scene, camera, controls, renderer, light;
// Assign Variables
```

```
//camera
camera = new THREE.PerspectiveCamera(60, window.innerWidth /
    window.innerHeight, 0.01, 1e10);
camera.position.z = -300;
//scene
scene = new THREE.Scene();
scene.add(camera);
//light
scene.add(new THREE.AmbientLight(0xf0f0f0));
light = new THREE.SpotLight(0xffffff, 1);
light.position.set(0, 1500, 200);
scene.add(light);
camera.add(light);
camera.add(light.target)
//renderer
renderer = new THREE.WebGLRenderer( { antialias: false } );
renderer.setClearColor( 0xf0f0f0 );
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
container = document.createElement( 'div' );
container.appendChild( renderer.domElement );
document.body.appendChild( container );
//stats
stats = new Stats();
container.appendChild( stats.dom );
```

A.3.3 On screen controls

```
<style>
    .navbar, .navbar.navbar-inverse {
```

```
        background: repeating-linear-gradient(160deg, #009688, rgba(240, 240, 240, 0) 160px, rgba(240, 240, 240, 0));
    }
    canvas {
        position: absolute;
        top:0;
        left:0;
    }
</style>

<!--nav bar-->
<nav class="navbar navbar-inverse" style="z-index:200; margin-bottom:0">
    <div class="container-fluid">
        <div class="navbar-header">
            <a href="#" on-click="loadParcels" class="navbar-left" style="padding-top: 10px; margin-left: -8px;margin-right: 5px;"><img src="../../public/Images/BiomediaLogo.png" height="40px" width="50px"></a>
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" on-click="loadParcels">Cortical Explorer</a>
        </div>
        <div class="collapse navbar-collapse" id="myNavbar">
            <ul class="nav navbar-nav">
                <li class="dropdown">
```

```
                    <a class="dropdown-toggle" data-toggle="dropdown" href="#">More
                        <span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        <li><a on-click="loadMyelin">Myelin</a></li>
                        <li><a on-click="loadThickness">Thickness</a></li>
                        <li><a on-click="loadSulc">Sulc</a></li>
                        <li><a on-click="loadFlat">Flat</a></li>
                        <li><a on-click="loadCurvature">Curvature</a></li>
                        <li><a on-click="loadCorrelation">Correlation</a></li>
                    </ul>
                </li>
            </ul>
            <!--<form class="navbar-form navbar-right">-->
                <!--<div class="form-group">-->
                    <!--<input type="text" class="form-control" placeholder="Search">-->
                <!--</div>-->
            <!--</form>-->
        </div>
    </div>
</nav>

<!--explode btn-->
<button
        type="button"
```

```
        id="explodeButton"
        class="btn"
        on-click="explode"
        style=" z-index:2;
                        position:absolute;
                        bottom:5px;
                        left:5px">
    Explode
</button>

<!--reoptimise labels btn-->
<button type="button"
id="reoptimiseButton"
class="btn"
on-click="reoptimiseLabels"
style=" z-index:2;
position:absolute;
bottom:5px;
left:105px" hidden="hidden">
    Shift labels
</button>

<!--reset btn -->
<button
        type="button"
        id="resetButton"
        class="btn"
        on-click="resetParcels"
        style=" z-index:2;
                        position:absolute;
```

```
                              bottom:45px;
                              left:5px;">
     Reset
</button>

<!--label toggle -->
<style>
     /* The switch - the box around the slider */
     .switch {
          position: relative;
          display: inline-block;
          width: 60px;
          height: 34px;
     }

     /* Hide default HTML checkbox */
     .switch input {display:none;}

     /* The slider */
     .slider {
          position: absolute;
          cursor: pointer;
          top: 0;
          left: 0;
          right: 0;
          bottom: 0;
          background-color: #ccc;
           -webkit-transition: .4s;
          transition: .4s;
     }
```

```
.slider:before {
    position: absolute;
    content: "";
    height: 26px;
    width: 26px;
    left: 4px;
    bottom: 4px;
    background-color: white;
    -webkit-transition: .4s;
    transition: .4s;
}

input:checked + .slider {
    background-color: #2196F3;
}

input:focus + .slider {
    box-shadow: 0 0 1px #2196F3;
}

input:checked + .slider:before {
    -webkit-transform: translateX(26px);
    -ms-transform: translateX(26px);
    transform: translateX(26px);
}

/* Rounded sliders */
.slider.round {
    border-radius: 34px;
```

```
    }

    .slider.round:before {
        border-radius: 50%;
    }
</style>
<div style=" z-index:2;
position:absolute;
bottom:95px;
left:15px;vertical-align: middle" >
    <label class="switch" style="margin: 0; padding: 0; bottom:
   0;">
        <input type="checkbox">
        <div class="slider round" on-click="toggleLabels"></div>
    </label>
    <button type="button" class="btn btn-sml btn-small" style="
   padding: 0; margin: 0 0 15px 0;">Labels</button>
</div>

<button type="button" style=" z-index:400;
position:absolute;
bottom: 140px;
right: 5px;" class="btn" on-click="morph">
    Inflate
</button>

<!--<button type="button" style=" z-index:2;-->
<!--position:absolute;-->
<!--top: 95px;-->
<!--right: 5px;" class="btn" on-click="reop">-->
```

```
    <!--reop-->
<!--</button>-->

<!--left/right toggles-->
<div style=" z-index:2;
position:absolute;
bottom:185px;
left:15px;vertical-align: middle">
    <label class="switch" style="margin: 0; padding: 0; bottom:
    0;">
        <input type="checkbox" checked>
        <div class="slider round" on-click="toggleLeft"></div>
    </label>
    <button type="button" class="btn btn-sml btn-small" style="
    padding: 0; margin: 0 0 15px 0;">Left</button>
</div>
<div style=" z-index:2;
position:absolute;
bottom:145px;
left:15px;vertical-align: middle" >
    <label class="switch" style="margin: 0; padding: 0; bottom:
    0;">
        <input type="checkbox" checked>
        <div class="slider round" on-click="toggleRight"></div>
    </label>
    <button type="button" class="btn btn-sml btn-small" style="
    padding: 0; margin: 0 0 15px 0;">Right</button>
</div>

<!--parcel info popup box-->
```

```
<div id= "infoBox" class="panel panel-primary" style="position: absolute; z-index: 250;top: 5px;right:5px;visibility: hidden; width:25%">
    <div class="panel-heading">
        <h3 class="panel-title">{{parcelName}}</h3>
        <button type="button" class="close" style="position: absolute; top:10px; right:15px;">
            <span aria-hidden="true" on-click="closePanel">&times;</span><span class="sr-only" >Close</span>
        </button>
    </div>
    <div class="panel-body">
        <table class="table table-striped table-hover" style="margin:0">
            <tbody>
            {{#if parcelDesc}}
            <tr>
                <td>description</td>
                <td>{{parcelDesc}}</td>
            </tr>
            {{/if}}
            {{#if isNew}}
            <tr>
                <td>new area?</td>
                <td>{{isNew}}</td>
            </tr>
            {{/if}}

            <tr>
                <td>Related Studies</td>
```

```
                <td>
                    {{#each keyStudyNames :i}}
                    <a href="{{keyStudyLinks[i]}}" target="
_blank">{{.}}</a>,
                    {{/each}}
                </td>
            </tr>

            {{#if otherNames}}
            <tr>
                <td>aka</td>
                <td>{{otherNames}}</td>
            </tr>
            {{/if}}
            <tr>
                <td>sections</td>
                <td>
                    {{#each sections :i}}
                    <button type="button"
                    class="btn" style="padding: 0px 5px 0px 5px;
 margin:0" on-click="@this.selectSection(i)">{{.}}</button>
                    {{/each}}
                </td>
            </tr>
            <tr>
                <td>top connections</td>
                <td>
                    <button type="button"
                            class="btn" style="padding: 0px 5px
0px 5px; margin:0" on-click="@this.showConnections(5)">
```

```
                        5
                    </button>
                    <button type="button"
                            class="btn" style="padding: 0px 5px
    0px 5px; margin:0" on-click="@this.showConnections(10)">
                        10
                    </button>
                    <button type="button"
                            class="btn" style="padding: 0px 5px
    0px 5px; margin:0" on-click="@this.showConnections(50)">
                        50
                    </button>
                </td>
            </tr>
            </tbody>
        </table>
        {{#if newMessageVisible}}
        <div
            style=" z-index:2;
                        height: 20px;
                        width:100%;
                        background:white;
                        margin:0;">
            *Yes, subarea of previous area.
        </div>
        {{/if}}
        <div
                id="colorToggle"
                on-click="toggleColors"
                style=" z-index:2;
```

```
                        height: 20px;
                        width:100%;
                        background:white;
                        margin:0;">
            <strong>move/colors</strong>
        </div>
    </div>
</div>

<!--quick view btns-->
<div class="btn-toolbar" style="z-index:20;position: absolute;
    bottom:5px;right:100px;margin:auto">
    <div class="btn-group">
        <a on-click="topview" class="btn">top</a>
        <a on-click="bottomview" class="btn">bottom</a>
        <a on-click="leftview" class="btn">left</a>
        <a on-click="rightview" class="btn">right</a>
        <a on-click="frontview" class="btn">front</a>
        <a on-click="backview" class="btn">back</a>
    </div>
</div>

<!--color key-->
<div class= "text-right" style="z-index:20;position: absolute;
    top:60px;left:5px;margin:auto;">
    <img src="../../public/Images/ColourKeyTransparent.png"
    height="200px" width="200px" alt="Color Key">
</div>

<!--credits-->
```

```html
<div class="container" style="z-index:20;position: absolute;
    bottom:50px;right:0px;margin:auto">
  <h3 class="text-right">Sam Budd, Emma Robinson, and Bernhard
    Kainz (Imperial, Dept. Computing)</h3>
  <p class="text-right">data from: Glasser MF, Van Essen D,
    Coalson TS, Robinson EC, Hacker CD, Harwell J,
      Yacoub E, Ugurbil K, Andersson J, Beckmann CF, Jenkinson M
    , Smith SM.
      A multi-modal parcellation of human cerebral cortex.
    Nature. 2016 Jul 20.
</div>
```

A.3.4 Adding parcel files to distribution

In *app.js* a Node package 'require' is used to include the models:

```javascript
for (let i = 0; i++; i < NUM_PARCELS) {
    require('../../public/models/ParcelsInflatedL/parcel' + i +
    '.vtk');
    require('../../public/models/ParcelsInflatedR/parcel' + i +
    '.vtk');
}
```

In *webpack.config.js* an importer that searches for *.vtk* files and loads them into the */dist* folder with a unique name is added:

```javascript
module.exports = {
    entry: {
        js: './app/js/app.js'
    },
```

```
    output: {
        filename: './dist/js/app-bundle.js'
    },
    module: {
        loaders: [
            {
                test: /\.js$/,
                exclude: /(node_modules|bower_components)/,
                loader: 'babel-loader',
                query: {
                    presets: ['es2015']
                }
            },
            {
                test: /\.html$/,
                loader: 'raw-loader'
            },
            {
                test: /\.vtk$/,
                exclude: /(public\/models\/ParcelsInflatedR)/,
                loader: 'file-loader?name=./dist/models/[name]L.[ext]'
            },
            {
                test: /\.vtk$/,
                exclude: /(public\/models\/ParcelsInflatedL)/,
                loader: 'file-loader?name=./dist/models/[name]R.[ext]'
            },
            {
```

```
                test: /\.ico$/,
                loader: 'raw-loader'
            }
        ]
    }
};
```

A.3.5 Highlighting objects

```
function highlightOnHover(INTERSECTED, intersects) {
    if ( intersects.length > 0 ) {
        if ( INTERSECTED) {
            INTERSECTED.material.emissive.setHex( INTERSECTED.
    currentHex );
        }
        document.body.style.cursor = 'pointer';
        INTERSECTED = intersects[ 0 ].object;
        INTERSECTED.currentHex = INTERSECTED.material.emissive.
    getHex();
        INTERSECTED.material.emissive.setHex( 0xff0000 );
    } else {
        if ( INTERSECTED ) {
            INTERSECTED.material.emissive.setHex( INTERSECTED.
    currentHex );
        }
        INTERSECTED = null;
        document.body.style.cursor = 'default';
    }
    return INTERSECTED;
```

```
}
```

A.3.6 Selecting object for drag

```
var SELECTED;
document.addEventListener( 'mousedown', onDocumentMouseDown,
    false );
document.addEventListener( 'mouseup', onDocumentMouseUp, false )
    ;

function onDocumentMouseDown(event) {
     raycaster.setFromCamera( mouse, camera );
     let intersects = raycaster.intersectObjects( intersectable )
    ;
     // prepare clicked on parcel to be dragged
     SELECTED = PARCEL_CONTROLS.selectParcel(intersects, scene,
    controls, SELECTED);
}

function onDocumentMouseUp(event) {
     SELECTED = null;
     PARCEL_CONTROLS.POSITION = new THREE.Vector3(0, 0, 0);
     PARCEL_CONTROLS.ORIGPOSITION = new THREE.Vector3(0, 0, 0);
     document.body.style.cursor = 'default';
     controls.enabled = true;
}
```

A.3.7 selectParcel()

```
function selectParcel(intersects, scene, controls, SELECTED) {
    if ( intersects.length > 0 ) {
        SELECTED = intersects[0].object;
        var name = SELECTED.name;

        this.POSITION = UTILS.getCenterPoint(SELECTED);
        this.ORIGPOSITION.x = this.POSITION.x;
        this.ORIGPOSITION.y = this.POSITION.y;
        this.ORIGPOSITION.z = this.POSITION.z;
        document.body.style.cursor = 'move';
        controls.enabled = false;
        var center;
        if (name.includes("L")) {
            var l = scene.getObjectByName("parcelL0");
            center = UTILS.getCenterPoint(l);
        } else {
            var r = scene.getObjectByName("parcelR0");
            center = UTILS.getCenterPoint(r);
        }
        this.LINE.x = this.POSITION.x - center.x;
        this.LINE.y = this.POSITION.y - center.y;
        this.LINE.z = this.POSITION.z - center.z;
        return SELECTED;
    } else {
        return null;
    }
}
```

Where $getCenterPoint(object)$ calculates the centre of the objects bounding box:

```
function getCenterPoint(mesh) {
        var middle = new THREE.Vector3();
        var geometry = mesh.geometry;
        geometry.computeBoundingBox();
        middle.x = (geometry.boundingBox.max.x + geometry.boundingBox.min.x) / 2;
        middle.y = (geometry.boundingBox.max.y + geometry.boundingBox.min.y) / 2;
        middle.z = (geometry.boundingBox.max.z + geometry.boundingBox.min.z) / 2;
        mesh.localToWorld( middle );
        return middle;
    }
```

A.3.8 Dragging parcel

A new variable is addded to each object as they are loaded that tracks how far the object has moved away from the brain:

```
mesh.extruded = 0;
```

this is updated in:

```
dragSelected: function (SELECTED, mouse) {
        if (mouse.y > this.PREVMOUSEY) {
          // mouse is moving upwards
            if (SELECTED.extruded < this.DRAGLIMIT) {
              // only move outwards of brain if parcel not dragged past limit already.
```

```
                // calculate offset of position
                this.POSITION.x = this.POSITION.x + (this.DRAGSCALE * this.LINE.x);
                this.POSITION.y = this.POSITION.y + (this.DRAGSCALE * this.LINE.y);
                this.POSITION.z = this.POSITION.z + (this.DRAGSCALE * this.LINE.z);
                // update position of parcel
                SELECTED.position.set(
                    this.POSITION.x - this.ORIGPOSITION.x,
                    this.POSITION.y - this.ORIGPOSITION.y,
                    this.POSITION.z - this.ORIGPOSITION.z);
                SELECTED.updateMatrix();
                // increment drag limit counter for parcel
                SELECTED.extruded++;
            }
        } else if (SELECTED.extruded > 0) {
          // only move downwards if parcel not in original position
            this.POSITION.x = this.POSITION.x - (this.DRAGSCALE * this.LINE.x);
            this.POSITION.y = this.POSITION.y - (this.DRAGSCALE * this.LINE.y);
            this.POSITION.z = this.POSITION.z - (this.DRAGSCALE * this.LINE.z);
            // update position of parcel
            SELECTED.position.set(
                this.POSITION.x - this.ORIGPOSITION.x,
                this.POSITION.y - this.ORIGPOSITION.y,
                this.POSITION.z - this.ORIGPOSITION.z);
```

```
            SELECTED.updateMatrix();
            // decrement drag limit counter for parcel
            SELECTED.extruded--;
        }
    }
```

A.3.9 Popout Parcel

Another variable is added to each parcel to keep track of whether the parcel is in its original position or not:

```
mesh.exploded = false;
```

The *popoutSelected* function then uses these variables to move the parcel correctly:

```
popoutSelected: function(SELECTED, scene) {
        let centerOfExplosion;
        let centerOfParcel;

        if (SELECTED.name.includes("R")) {
            centerOfExplosion = UTILS.getCenterOfExplosionR(
    scene);
        } else {
            centerOfExplosion = UTILS.getCenterOfExplosionL(
    scene);
        }
        let ghost = scene.getObjectByName("ghost" + SELECTED.
    name.split('l')[1]);
```

```

    if (SELECTED.exploded && (SELECTED.extruded == this.POPLIMIT)) {
        centerOfParcel = SELECTED.position;
        UTILS.moveAlong(SELECTED, [centerOfParcel, centerOfExplosion], {});
        SELECTED.exploded = false;
        SELECTED.extruded = 0;
        //ghost
        ghost.visible = false;

    } else if (SELECTED.extruded == 0){
        centerOfParcel = UTILS.getCenterPoint(SELECTED);
        UTILS.moveAlong(SELECTED, [centerOfExplosion, centerOfParcel], {});
        SELECTED.exploded = true;
        SELECTED.extruded++;
        SELECTED.prevCenter = centerOfParcel;
        //ghost
        ghost.visible = true;
    } else {
        centerOfParcel = UTILS.getCenterPoint(SELECTED);
        UTILS.moveAlong(SELECTED, [SELECTED.prevCenter, centerOfParcel], {});
        SELECTED.prevCenter = centerOfParcel;
        SELECTED.extruded++;
        //ghost
        ghost.visible = true;

    }
```

```
    }
```

The function *moveAlong* that is called in *popoutSelected* uses Tween to handle the smooth animation of the parcel. *moveAlong* builds a 'Tween' object that is updated on each call to the existing *render* function:

```
moveAlong: function (object, shape, options) {
        options = _.merge({
            from: 0,
            to: 1,
            duration: 10,
            speed: 50,
            start: true,
            yoyo: false,
            onStart: null,
            onComplete: this.noop,
            onUpdate: this.noop,
            smoothness: 100,
            easing: TWEEN.Easing.Quadratic.In
        }, options);

        // array of vectors to determine shape
        if (shape instanceof THREE.Shape) {
            //Nothing to do here
        } else if ( shape.constructor === Array ) {
            shape = new THREE.CatmullRomCurve3(shape);
        } else {
            throw '2nd argument is not a Shape, nor an array of vertices';
        }
```

```
    options.duration = options.duration || shape.getLength();
    options.length = options.duration * options.speed;

    var tween = new TWEEN.Tween({ distance: options.from })
        .to({ distance: options.to }, options.length)
        .easing( options.easing )
        .onStart( options.onStart )
        .onComplete( options.onComplete )
        .onUpdate(function(){
            // get the position data half way along the path
            var pathPosition = shape.getPointAt( this.distance );
            // move to that position
            object.position.set( pathPosition.x, pathPosition.y, pathPosition.z );

            object.updateMatrix();

            if ( options.onUpdate ) { options.onUpdate( this, shape ); }
        })
        .yoyo( options.yoyo );

    if ( options.yoyo ) {
        tween.repeat( Infinity );
    }

    if ( options.start ) { tween.start(); }
```

```

        return tween;
    }
```

and in *render* add the line:

```
TWEEN.update();
```

A.3.10 MongoDB Setup

To set up a MongoDB instance the necessary files from the Mongo website[22] were installed. The following command starts the instance:

```
./mongod --dbpath /Users/sambudd/Imperial/Individual\ Project/
    cortical-back/data/
```

This starts the MongoDB instance. The instance listens for connections on port 27017. MongoDB is the 'Database' as shown in Figure 13. With the instance running the relevant data is loaded into the database with the following command:

```
./mongoimport -d mydb -c parcels --type csv --file ../../
    cortical-back/dataWithLinks.csv --headerline
```

A.3.11 API setup

Another NodeJS project is set up, using the 'Express' [8].

This API listens for connections on port 3000 and serves responses containing data from the MongoDB instance:

```
var express = require('express'),
    app = express(),
    port = process.env.PORT || 3000,
    bodyParser = require('body-parser');
    cors = require('cors');
require('jquery');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(cors());

var routes = require('./api/routes/parcelRoutes');
routes(app);

app.listen(port);
```

This *server.js* then passes responsibility of handling connections to *parcelRoutes.js*:

```
let parcelController = require('../controllers/parcelController'
    );
app.route('/parcels/names').get(parcelController.
    list_all_parcels);
app.route('/parcels/:parcelIndex').get(parcelController.
    find_parcel);
app.route('/parcels/sections/:sectionIndex').get(
    parcelController.find_sections);
```

Where *parcelController* queries the MongoDB instance and returns the requested information:

```
let mongodb = require('mongodb');
const MongoClient = mongodb.MongoClient;
const url = 'mongodb://localhost:27017/mydb';

let DB;
MongoClient.connect(url, function(err, db) {
    if (err) {
        console.log("Couldnt Connect");
    } else {
        console.log("Connected to Mongo");
        DB = db;
    }
});

exports.list_all_parcels = function(req, res) {
    var collection = DB.collection('parcels');
    collection.find({}).toArray(function(err, result) {
        if (err) {
            console.log(err)
        } else if (result.length) {
            res.json(result);
        }
    })
};

exports.find_parcel = function(req, res) {
    var collection = DB.collection('parcels');
    collection.find({"Parcel Index": Number(req.params.
```

```
    parcelIndex)}).toArray(function(err, result) {
        if (err) {
            console.log("err")
        } else if (result) {
            res.json(result);
        }
    })
};

exports.find_sections = function (req, res) {
    var collection = DB.collection('parcels');
    console.log(req.params.sectionIndex);
    var foundEntries = [];
    collection.find( {$or: [
            {"Sections": {$regex: ".*"+ req.params.sectionIndex
    + ".*"}},
            {"Sections": Number(req.params.sectionIndex)}]}
    ).toArray(function(err, result) {
        if (err) {
            console.log(err)
        } else if (result) {
            result.forEach((element) => {
                var sec = element["Sections"];
                console.log(sec);
                if (sec.length) {
                    var sections = sec.split(',');
                    sections.forEach((section) => {
                        if (section == req.params.sectionIndex)
    {
                            foundEntries.push(element["Parcel
```

```
    Index"])
                        }
                    })
                } else {
                    foundEntries.push(element["Parcel Index"])
                }
            });
            res.json(foundEntries);
        }
    })
};
```

A.3.12 Information box

```
var App = new Ractive({
    el: '#app',
    template: html,
    data: {
        parcelIndex: 0,
        parcelName: "",
        parcelDesc: "",
        isNew: "",
        newMessageVisible: false,
        keyStudies: "",
        otherNames: "",
        sections: [],
        keyStudyNames: [],
        keyStudyLinks: []
    },
```

```
    selectSection: function(i) {
        explodeSections(this.get("sections")[i]);
    },

    showConnections: function(n) {
        if (this.get("parcelName").includes('R')) {
            explodeConnections(this.get("parcelIndex") + 181, n);
        } else {
            explodeConnections(this.get("parcelIndex"), n);
        }

    }
});
```

The variables in 'data' are bound to html elements - they always display whatever is stored in the 'data' variable:

```
<!--parcel info popup box-->
<div id= "infoBox" class="panel panel-primary" style="position: absolute; z-index: 250;top: 5px;right:5px;visibility: hidden; width:25%">
    <div class="panel-heading">
        <h3 class="panel-title">{{parcelName}}</h3>
        <button type="button" class="close" style="position: absolute; top:10px; right:15px;">
            <span aria-hidden="true" on-click="closePanel">&times;</span><span class="sr-only" >Close</span>
        </button>
    </div>
```

```
    <div class="panel-body">
        <table class="table table-striped table-hover" style="margin:0">
            <tbody>
            {{#if parcelDesc}}
            <tr>
                <td>description</td>
                <td>{{parcelDesc}}</td>
            </tr>
            {{/if}}
            {{#if isNew}}
            <tr>
                <td>new area?</td>
                <td>{{isNew}}</td>
            </tr>
            {{/if}}

            <tr>
                <td>Related Studies</td>
                <td>
                    {{#each keyStudyNames :i}}
                    <a href="{{keyStudyLinks[i]}}" target="_blank">{{.}}</a>,
                    {{/each}}
                </td>
            </tr>

            {{#if otherNames}}
            <tr>
                <td>aka</td>
```

```
                <td>{{otherNames}}</td>
            </tr>
            {{/if}}
            <tr>
                <td>sections</td>
                <td>
                    {{#each sections :i}}
                    <button type="button"
                    class="btn" style="padding: 0px 5px 0px 5px; margin:0" on-click="@this.selectSection(i)">{{.}}</button>
                    {{/each}}
                </td>
            </tr>
            <tr>
                <td>top connections</td>
                <td>
                    <button type="button"
                            class="btn" style="padding: 0px 5px 0px 5px; margin:0" on-click="@this.showConnections(5)">
                        5
                    </button>
                    <button type="button"
                            class="btn" style="padding: 0px 5px 0px 5px; margin:0" on-click="@this.showConnections(10)">
                        10
                    </button>
                    <button type="button"
                            class="btn" style="padding: 0px 5px 0px 5px; margin:0" on-click="@this.showConnections(50)">
                        50
```

```
                    </button>
                </td>
            </tr>
            </tbody>
        </table>
        {{#if newMessageVisible}}
        <div
            style=" z-index:2;
                        height: 20px;
                        width:100%;
                        background:white;
                        margin:0;">
            *Yes, subarea of previous area.
        </div>
        {{/if}}
        <div
                id="colorToggle"
                on-click="toggleColors"
                style=" z-index:2;
                        height: 20px;
                        width:100%;
                        background:white;
                        margin:0;">
            <strong>move/colors</strong>
        </div>
    </div>
</div>
```

A.3.13 Getting sections

API call added to 'API':

```
getParcelsForSection: function (section, callback) {
        http.get({
            hostname: this.API_ADDR,
            port: this.API_PORT,
            path: '/parcels/sections/' + section,
            agent: true
        }, (res) => {
            res.on('data', (data) => {
                let jsonObject = JSON.parse(data);
                callback(new Set(jsonObject));
            })
        });
    }
```

Calling function:

```
function explodeSections(section) {
    API.getParcelsForSection(section, (uniq) => {
        if (highlightWithColors) {
            whiteOutAllParcels();
            PARCEL_CONTROLS.recolourParcels(uniq, scene)
        } else {
            PARCEL_CONTROLS.explodeParcels(uniq, scene);
        }
    });
}
```

A.3.14 Getting strongest connections

```
con = load('GLASS_PRO_360_PCORR.mat');
mat = con.connectivity;

avgCon = zeros(360, 360);
for j=1:360
    for k=1:360
        for i=1:100
            avgCon(j, k) = avgCon(j, k) + mat(j, k, i);
            if (i==100)
                avgCon(j, k) = avgCon(j, k)/100;
            end
        end
    end
end

weights = zeros(360);
indices = zeros(360);

for i=1:360
    [weight, index] = sort(avgCon(i,:), 'descend');
    weights(i,:) = weight;
    indices(i,:) = index;
end

%Write to json

%Parcel Index List
names = indices(:, 1);
%Connected Index List
```

```
connections = indices;
conStrengths = weights;
json = jsonencode(table(names, connections, conStrengths));

fid = fopen('connections.json','wt');
fprintf(fid, '%s', json);
fclose(fid);
```

The file is then loaded into the MongoDB instance:

```
/mongoimport -d mydb -c connections --type json --file ../../cortical-back/connections.json --jsonArray
```

Routes to access data via API:

In front end:

```
function explodeConnections(parcelIndex, numConnections) {
    API.getConnectionsForParcel(parcelIndex, numConnections, (data) => {
        var parcelList = data["connections"];
        var weightList = data["conStrengths"];
        var parcelNames = [];
        for (var i = 0; i < numConnections; i++) {
            if (parcelList[i] > 180) {
                parcelNames[i] = ('parcelR' + (parcelList[i] - 180));
            } else {
                parcelNames[i] = ('parcelL' + parcelList[i]);
            }
        }
```

```
        if (highlightWithColors) {
            whiteOutAllParcels();
            PARCEL_CONTROLS.recolourParcelsByName(parcelNames,
    scene);
        } else {
            PARCEL_CONTROLS.explodeParcelsByName(parcelNames,
    scene);
        }
    })
}
```

in *parcelRoutes.js*:

```
//Connections (Left Hemisphere indexed first)
    app.route('/connections/:parcelIndex/:numReturned')
        .get(parcelController.get_connections);
```

in *parcelController.js*

```
exports.get_connections = function(req, res) {
    var collection = DB.collection('connections');
    collection.find({"names": Number(req.params.parcelIndex)}).
    toArray(function (err, result) {
        if (err) {
            console.log("err")
        } else if (result) {
            result.forEach((parcel) => {
                parcel["connections"] = parcel["connections"].
    slice(0, req.params.numReturned);
```

```
                parcel["conStrengths"] = parcel["conStrengths"].
    slice(0, req.params.numReturned);
            });
            res.json(result);
        }
    })
};
```

A.3.15 Explosion diagram

```
<!--explode btn-->
<button
        type="button"
        id="explodeButton"
        class="btn"
        on-click="explode"
        style=" z-index:2;
                        position:absolute;
                        bottom:5px;
                        left:5px">
    Explode
</button>
```

```
App.on('explode', () => {
        PARCEL_CONTROLS.explode(scene, NUM_PARCELS);
    });
```

```
explode: function (scene, NUM_PARCELS) {
        var allIn = true;
        let centerOfExplosionL = UTILS.getCenterOfExplosionL(scene);
        let centerOfExplosionR = UTILS.getCenterOfExplosionR(scene);
        for (let i = 1; i < NUM_PARCELS; i ++) {
            if (!this.LEFT_HIDDEN) {
                let centerOfParcelL;
                let parcelL = scene.getObjectByName("parcelL" + i);
                let ghost = scene.getObjectByName("ghostL" + i);
                if (parcelL.exploded && (parcelL.extruded == this.POPLIMIT)) {
                    centerOfParcelL = parcelL.position;
                    UTILS.moveAlong(parcelL, [centerOfParcelL, centerOfExplosionL], {});
                    parcelL.exploded = false;
                    parcelL.extruded = 0;
                    allIn = allIn && true;
                    document.getElementById("resetButton").disabled = null;

                    ghost.visible = false;

                } else if (parcelL.extruded == 0){
                    centerOfParcelL = UTILS.getCenterPoint(parcelL);
                    UTILS.moveAlong(parcelL, [centerOfExplosionL, centerOfParcelL], {});
```

```
                parcelL.exploded = true;
                parcelL.extruded++;
                parcelL.prevCenter = centerOfParcelL;
                allIn = false;
                document.getElementById("resetButton").disabled = 'true';


                ghost.visible = true;
            } else {
                centerOfParcelL = UTILS.getCenterPoint(parcelL);
                UTILS.moveAlong(parcelL, [parcelL.prevCenter, centerOfParcelL], {});
                parcelL.prevCenter = centerOfParcelL;
                parcelL.extruded++;
                allIn = false;

                ghost.visible = true;
            }
        }

        if (!this.RIGHT_HIDDEN) {
            let centerOfParcelR;
            let parcelR = scene.getObjectByName("parcelR" + i);
            let ghost = scene.getObjectByName("ghostR" + i);

            if (parcelR.exploded && (parcelR.extruded == this.POPLIMIT)) {
```

```
                    centerOfParcelR = parcelR.position;
                    UTILS.moveAlong(parcelR, [centerOfParcelR, centerOfExplosionR], {});
                    parcelR.exploded = false;
                    parcelR.extruded = 0;
                    allIn = allIn && true;
                    ghost.visible = false;
                } else if (parcelR.extruded == 0){
                    centerOfParcelR = UTILS.getCenterPoint(parcelR);
                    UTILS.moveAlong(parcelR, [centerOfExplosionR, centerOfParcelR], {});
                    parcelR.exploded = true;
                    parcelR.extruded++;
                    allIn = false;
                    parcelR.prevCenter = centerOfParcelR;

                    ghost.visible = true;
                } else {
                    centerOfParcelR = UTILS.getCenterPoint(parcelR);
                    UTILS.moveAlong(parcelR, [parcelR.prevCenter, centerOfParcelR], {});
                    parcelR.prevCenter = centerOfParcelR;
                    parcelR.extruded++;
                    allIn = false;

                    ghost.visible = true;
                }
            }
```

```
        }
    }
```

A.3.16 Viewing Collections

Explosion

```
explodeParcels: function (parcelIds, scene) {
        var allIn = true;
        let centerOfExplosionL = UTILS.getCenterOfExplosionL(
    scene);
        let centerOfExplosionR = UTILS.getCenterOfExplosionR(
    scene);
        parcelIds.forEach((i) => {
            let centerOfParcelL;
            let parcelL = scene.getObjectByName("parcelL" + i);
            var ghost = scene.getObjectByName("ghostL" + i);
            if (parcelL.exploded && (parcelL.extruded == this.
    POPLIMIT)) {
                centerOfParcelL = parcelL.position;
                UTILS.moveAlong(parcelL, [centerOfParcelL,
    centerOfExplosionL], {});
                parcelL.exploded = false;
                parcelL.extruded = 0;
                allIn = true && allIn
                //ghost
                ghost.visible = false;

            } else if (parcelL.extruded == 0){
```

```
                centerOfParcelL = UTILS.getCenterPoint(parcelL);
                UTILS.moveAlong(parcelL, [centerOfExplosionL,
centerOfParcelL], {});
                parcelL.exploded = true;
                parcelL.extruded++;
                parcelL.prevCenter = centerOfParcelL;
                allIn = false;
                //ghost
                ghost.visible = true;
            } else {
                centerOfParcelL = UTILS.getCenterPoint(parcelL);
                UTILS.moveAlong(parcelL, [parcelL.prevCenter,
centerOfParcelL], {});
                parcelL.prevCenter = centerOfParcelL;
                parcelL.extruded++;
                allIn = false;
                //ghost
                ghost.visible = true;
            }

            let centerOfParcelR;
            let parcelR = scene.getObjectByName("parcelR" + i);
            ghost = scene.getObjectByName("ghostR" + i);
            if (parcelR.exploded && (parcelR.extruded == this.
POPLIMIT)) {
                centerOfParcelR = parcelR.position;
                UTILS.moveAlong(parcelR, [centerOfParcelR,
centerOfExplosionR], {});
                parcelR.exploded = false;
                parcelR.extruded = 0;
```

```
                allIn = true && allIn

                ghost.visible = false;

            } else if (parcelR.extruded == 0){
                centerOfParcelR = UTILS.getCenterPoint(parcelR);
                UTILS.moveAlong(parcelR, [centerOfExplosionR,
centerOfParcelR], {});
                parcelR.exploded = true;
                parcelR.extruded++;
                parcelR.prevCenter = centerOfParcelR;
                allIn = false;

                ghost.visible = true;
            } else {
                centerOfParcelR = UTILS.getCenterPoint(parcelR);
                UTILS.moveAlong(parcelR, [parcelR.prevCenter,
centerOfParcelR], {});
                parcelR.prevCenter = centerOfParcelR;
                parcelR.extruded++;
                allIn = false;

                ghost.visible = true;
            }
        });
        if (this.DEFAULT && allIn) {
            document.getElementById("resetButton").disabled = '
true';
        } else {
            document.getElementById("resetButton").disabled =
```

```
    null;
        }
    }
```

Recolouring

First all parcels are 'whited' out:

```
function whiteOutAllParcels() {
    for (var i = 1; i < NUM_PARCELS; i ++) {
        var parcelL = scene.getObjectByName("parcelL" + i);
        var parcelR = scene.getObjectByName("parcelR" + i);
        parcelL.material.emissive.setHex( 0xffffff );
        parcelL.material.emissiveIntensity = 0.75;
        parcelR.material.emissive.setHex( 0xffffff );
        parcelR.material.emissiveIntensity = 0.75;
    }
}
```

The parcels that belong to the collection are then reset to their default appearance.

```
recolourParcels: function (parcelIds, scene) {
        parcelIds.forEach((i) => {
            let parcelL = scene.getObjectByName("parcelL" + i);
            let parcelR = scene.getObjectByName("parcelR" + i);
            parcelL.material.emissive.setHex(0x000000);
            parcelR.material.emissive.setHex(0x000000);
        });
```

```
    },
```

A.3.17 Labels

```
//initialises smart labels
    smartLabels: function(scene, camera, renderer, NUM_PARCELS,
   LEFT_HIDDEN, RIGHT_HIDDEN) {
        this.scene = scene;
        this.NUM_PARCELS = NUM_PARCELS;
        this.camera = camera;
        this.renderer = renderer;

        //INITIATE SMART LABELS
        for (let i = 1; i < this.NUM_PARCELS; i++) {
            API.getInfoForParcel(i, (parcelInfo) => {
                if (LEFT_HIDDEN || RIGHT_HIDDEN) {
                    this.CURRENT_CENTER_LEFT = this.
   LEFTHEMICENTER;
                    this.CURRENT_CENTER_RIGHT = this.
   RIGHTHEMICENTER
                } else {
                    this.CURRENT_CENTER_LEFT = this.CENTER;
                    this.CURRENT_CENTER_RIGHT = this.CENTER;
                }
                this.addLabelFromCenter(this.scene.
   getObjectByName("parcelL" + i), this.scene, parcelInfo["Area
   Name"], this.CURRENT_CENTER_LEFT);
                this.addLabelFromCenter(this.scene.
   getObjectByName("parcelR" + i), this.scene, parcelInfo["Area
```

```
    Name"], this.CURRENT_CENTER_RIGHT);
                    if (i == this.NUM_PARCELS - 1) {
                        console.log("optimise");
                        this.smartLoaded = true;
                        // this.optimiseLabels();
                    }
                });
            }
        },
```

Calculate label pole start point by calculating the closest vertex to the objects centre:

```
getLineStartPointForParcel: function(parcel) {
        var minDist = 1000000;
        // var maxDist = -1;
        parcel.labelStartVertexIndex = 0;
        var center = UTILS.getCenterPoint(parcel);
        for (let i = 0; i < parcel.geometry.vertices.length; i
    ++) {
            let dist = parcel.geometry.vertices[i].distanceTo(
    center);
            if (dist < minDist) {
                minDist = dist;
                parcel.labelStartVertexIndex = i;
            }
        }
        return parcel.geometry.vertices[parcel.
    labelStartVertexIndex];
```

```
    }
```

Create text sprite:

```
makeTextSprite: function (message, parameters) {
        if ( parameters === undefined ) parameters = {};

        var fontface = parameters.hasOwnProperty("fontface") ?
            parameters["fontface"] : "Arial";

        var fontsize = parameters.hasOwnProperty("fontsize") ?
            parameters["fontsize"] : 80;

        var borderThickness = parameters.hasOwnProperty("borderThickness") ?
            parameters["borderThickness"] : 0;

        var borderColor = parameters.hasOwnProperty("borderColor") ?
            parameters["borderColor"] : { r:0, g:0, b:0, a:0.2 };

        var backgroundColor = parameters.hasOwnProperty("backgroundColor") ?
            parameters["backgroundColor"] : { r:255, g:255, b:255, a:0.2 };

        var canvas = document.createElement('canvas');
        var context = canvas.getContext('2d');

```

```
        // var size = 0;
        canvas.width = 300;
        canvas.height = 100;
        context.textAlign = "center";
        context.fillStyle = "rgba(255, 255, 255, 1)";
        context.fillRect( 0, 0, 300, 100 );

        context.font = "Bold " + fontsize + "px " + fontface;

        // background color
        context.fillStyle   = "rgba(" + backgroundColor.r + "," + backgroundColor.g + ","
            + backgroundColor.b + "," + backgroundColor.a + ")";
        // border color
        context.strokeStyle = "rgba(" + borderColor.r + "," + borderColor.g + ","
            + borderColor.b + "," + borderColor.a + ")";

        context.lineWidth = borderThickness;
        // 1.4 is extra height factor for text below baseline: g,j,p,q.
        // var metrics = context.measureText( message );
        // var textWidth = metrics.width;
        // this.roundRect(context, borderThickness/2, borderThickness/2, textWidth + borderThickness, fontsize * 1.4 + borderThickness, 1);

        // text color
        context.fillStyle = "rgba(0, 0, 0, 1)";

```

```
    context.fillText( message, canvas.width/2, fontsize + borderThickness);

    // canvas contents will be used for a texture
    var texture = new THREE.Texture(canvas);
    texture.needsUpdate = true;

    var spriteMaterial = new THREE.SpriteMaterial({ map: texture, transparent: false});

    var sprite = new THREE.Sprite( spriteMaterial );
    sprite.scale.set(12, 4,1);
    return sprite;
}
```

B Parcel Metadata

Parcel Index	Area Name	Area Description	New?	Sections	Other Names	Key Studies	Links To Studies
1	V1	Primary Visual Cortex	No	1,2	17, hOC1, OC, BA17	Amunts et al 2000[52], Fischl et al 2008[65], Abdollahi et al 2014[48]	https://www https://www https://www

2	MST	Medial SuperiorTemporal Area	No	5,15	MSTv, hOC5, hOC5v	Abdollahi et al 2014, Kolster et al 2010[84], Malikovic et al 2007[89], Fischl et al 2008	https://www https://www https://www https://www
3	V6	Sixth Visual Area	No	2,3,18	112	Pitzalis et al 2006[101], Pitzalis et al 2013[102], Sereno et al 2012[110], Nieuwenhuys et al 2014[95]	https://www https://www https://www https://www
4	V2	Second Visual Area	No	1,2	18, hOC2, OB, BA18	Amunts et al 2000, Fischl et al 2008, Schira et al 2009[109], Abdollahi et al 2014, Wang et al 2015[126], Wandell and Winawer 2011/citeWandell2011ImagingBrain	https://www https://www https://www https://www https://www https://www
5	V3	Third Visual Area	No	2	V3d, V3v, VP,hOC3d, hOC3v	Abdollahi et al 2014, Rottschy et al 2007[105], Schira et al 2009, Kujovic et al 2012[86], Wang et al 2015, Wandell and Winawer 2011	https://www https://www https://www https://www https://www https://www

6	V4	Fourth Visual Area	No	2,3,4,5	V4d, V4v, hV4,hOC4v, hOC4lp, LO1	Hansen et al 2007[78], Abdollahi et al 2014, Rottschy et al 2007, Malikovic et al 2015[90]	https://www https://www https://www , https://www
7	V8	Eighth Visual Area	No	2,4,5	VO1	Hadjikhani et al 1998[76], Abdollahi et al 2014	https://www https://www
8	4	Primary Motor Cortex	No	6,7,8,9	BA4, 4a, 4p, M1, PMC, F1	Fischl et al 2008, Geyer et al 1996[68]	https://www https://www
9	3b	Primary Senso-ryCortex	No	6,7,9	S1, 3	Fischl et al 2008, Geyer et al 1999[69], Geyer et al 2000[70]	https://www https://www https://www
10	FEF	Frontal Eye Fields	No	6,8,22		Glasser and Van Essen 2011[71], Amiez and Petrides 2009[49]	https://www https://www
11	PEF	Premotor Eye Field	No	6,8,21,22	6v2	Amiez and Petrides 2009, Amunts et al 2010[51]	https://www https://www
12	55b	Area 55b	No	6,8,22		Hopf 1956 [81]	https://www

13	V3A	Area V3A	No	2,3	V3D, hOC4d	Abdollahi et al 2014, Swisher et al 2007[112], Kujovic et al 2012[86], Wandell and Winawer 2011, Larsson and Heeger 2006[88], Tootell et al 1997[115]	https://www https://www https://www https://www https://www https://www
14	RSC	RetroSplenialComplex	No	13,18	29,30	Glasser and Van Essen 2011, Vogt 2009[124], Palomero-Gallagher et al 2009[98]	https://www https://glob neurobiology and- disease- 97801985669 https://www
15	POS2	Parieto-OccipitalSulcus Area 2	Yes*	16,18		Glasser and Van Essen 2011	https://www
16	V7	Seventh Visual Area	No	3	IPS0	Abdollahi et al 2014, Swisher et al 2007, Larsson and Heeger 2006, Tootell et al 1998[114], Hagler et al 2007[77],Wang et al 2015	https://www https://www https://www https://www https://www https://www

17	IPS1	IntraParietal SulcusArea 1	No	3,16,17		Swisher et al 2007, Wang et al 2015, Hagler et al 2007	https://www https://www , https://www
18	FFC	Fusiform FaceComplex	No	4,5,14	FFA, FG2	Glasser and Van Essen 2011, Kanwisher and Yovel 2006[82], Caspers et al 2013[57], Weiner et al 2014[127]	https://www https://www https://www https://www
19	V3B	Area V3B	No	3,5,17	V3C	Abdollahi et al 2014, Larsson and Heeger 2006,Swisher et al 2007, Wandell and Winawer 2011,Smith et al 1998[111]	https://www https://www https://www https://www https://www
20	LO1	Area Lateral Occipital1	No	2,5	LO2, hOC4la	Abdollahi et al 2014, Hansen et al 2007, Malikovic et al 2015[90], Larsson and Heeger 2006	https://www https://www https://www https://www
21	LO2	Area Lateral Occipital2	No	2,4,5	LO1, hOC4la	Abdollahi et al 2014, Hansen et al 2007, Malikovic et al 2015, Larsson and Heeger 2006	https://www https://www https://www https://www

22	PIT	PosteriorInferoTemporal	No	2,4,5	phPITv, ph-PITd,OFA, hOC4la	Abdollahi et al 2014, Kolster et al 2010, Malikovic et al 2015, Kanwisher and Yovel 2006, Tsao et al 2008[118]	https://www https://www https://www https://www https://www
23	MT	Complex Middle Temporal Area	No	5,15	hOC5, hOC5d	Abdollahi et al 2014, Kolster et al 2010, Malikovic et al 2007, Fischl et al 2008	https://www https://www https://www https://www
24	A1	Primary Audito-ryCortex	No	10	Core, R1, TC, TE1.0,TE1, 41	Glasser and Van Essen 2011, Moerel et al 2014[91], von Economo and Koskinas 1925[117], Triarhou 2007[116],Morosan et al 2001[93]	https://www https://www https://www , https://www https://www
25	PSL	PeriSylvian Lan-guageArea	Yes	9,10,11,15,17			
26	SFL	Superior Frontal-Language Area	Yes	7,19,22			
27	PCV	PreCuneus VisualArea	No	7,16,18	PrCu	Sereno et al 2012[110]	https://www

28	STV	Superior Tempo-ralVisual Area	Yes	11,15,17				
29	7Pm	Medial Area 7P	Yes	16,18	7P	Scheperjans et al 2008a[107], Scheperjans et al 2008b [108]	https://www https://www	
30	7m	Area 7m	No	16,18		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www	
31	POS1	Parieto-OccipitalSulcus Area 1	Yes*	18	”RetrosplenialCortex”	Glasser and Van Essen 2011	https://www	
32	23d	Area 23d	No	18,19		Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www	
33	v23ab	Area ventral 23 a+b	No	18	23a, 23b, v23	Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www	

34	d23ab	Area dorsal 23 a+b	No	18	23a, 23b, d23	Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www
35	31pv	Area 31p ventral	Yes*	18	31, 31d, 31v	Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www
36	5m	Area 5m	No	6,7		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
37	5mv	Area 5m ventral	Yes*	7,16,18	5ci	Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
38	23c	Area 23c	No	7,18,19		Vogt, 2009 Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www
39	5L	Area 5L	No	6,7,16		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www

40	24dd	Dorsal Area 24d	No	6,7,18	24d	Palomero-Gallagher et al 2009, Vogt and Vogt 2003[123]	https://www http://www
41	24dv	Ventral Area 24d	No	7,19	24d	Palomero-Gallagher et al 2009, Vogt and Vogt 2003	https://www http://www
42	7AL	Lateral Area 7A	Yes*	6,7,16		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
43	SCEF	Supplementary andCingulate Eye Field	Yes*	7,19,22	SEF, CEF, 6, SMA,SMAr	Amiez and Petrides 2009	https://www
44	6ma	Area 6m anterior	Yes*	7,8,22	SMAr, 6, SMA	Fischl et al 2008, Vorobiev et al 1998[125], Geyer 2004[67]	https://www https://www http://www
45	7Am	Medial Area 7A	Yes*	7,16,18		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
46	7Pl	Lateral Area 7P	Yes*	16,18		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
47	7PC	Area 7PC	No	6,16		Scheperjans et al 2008a, Scheperjans et al 2008b	https://www https://www
48	LIPv	Area LateralIntraParietal ventral	Yes*	16	hIP3	Van Essen et al 2012a[120], Scheperjans et al 2008a,Scheperjans et al 2008b	https://www https://www https://www

49	VIP	Ventral IntraPari-etalCom-plex	Yes*	16		Van Essen et al 2012a	https://www
50	MIP	Medial IntraPari-etalArea	Yes*	3,16,17		Van Essen et al 2012a	https://www
51	1	Area 1	No	6,7,9,17		Fischl et al 2008, Geyer et al 1999, Geyer et al 2000	https://www https://www https://www
52	2	Area 2	No	6,7,16,17		Fischl et al 2008, Grefkes et al 2000[75]	https://www https://www
53	3a	Area 3a	No	6,7,9,17		Fischl et al 2008, Geyer et al 1999, Geyer et al 2000	https://www https://www https://www
54	6d	Dorsal area 6	Yes*	6,7,8	6, 6a_	Fischl et al 2008, Geyer 2004, Geyer et al 2000	https://www http://www https://www
55	6mp	Area 6mp	Yes*	6,7,8	SMAc, 6, SMA	Fischl et al 2008, Vorobiev et al 1998, Geyer 2004	https://www https://www http://www
56	6v	Ventral Area 6	No	6,8,9	6, 6v1	Fischl et al 2008, Amunts et al 2010, Geyer 2004	https://www https://www http://www

57	p24pr	Area Posterior 24prime	No	7,18,19	p24’	Vogt 2009	https://glob neurobiology and-disease-97801985669
58	33pr	Area 33 prime	No	18,19	33’, 16	Vogt 2009, Nieuwenhuys et al 2014[95]	https://glob neurobiology and-disease-97801985669 https://www
59	a24pr	Anterior 24 prime	No	19	a24’	Vogt 2009	https://glob neurobiology and-disease-97801985669
60	p32pr	Area p32 prime	Yes*	7,19	32’	Vogt 2009	https://glob neurobiology and-disease-97801985669

61	a24	Area a24	Yes*	19	24, s24	Vogt 2009, Palomero-Gallagher et al 2015[97]	https://glob neurobiology and- disease- 97801985669 https://www
62	d32	Area dorsal 32	No	19	32	Vogt 2009	https://glob neurobiology and- disease- 97801985669
63	8BM	Area 8BM	Yes*	7,19,22	8B	Petredes and Pandya 1999[100]	https://www
64	p32	Area p32	No	19,20	32ac, 32	Van Essen et al 2012b[119],Ongur et al 2003, Vogt 2009[124],Palomero-Gallagher et al 2009	https://www https://www https://glob neurobiology and- disease- 97801985669 https://www
65	10r	Area 10r	Yes*	19,20		Van Essen et al 2012b, Ongur et al 2003[96]	https://www https://www

66	47m	Area 47m	No	20,21		Van Essen et al 2012b, Ongur et al 2003, Glasser and Van Essen 2011	https://www https://www https://www
67	8Av	Area 8Av	Yes*	8,22		Petredes and Pandya 1999	https://www
68	8Ad	Area 8Ad	Yes*	22		Petredes and Pandya 1999	https://www
69	9m	Area 9 Middle	Yes*	19,20,22	9	Petredes and Pandya 1999	https://www
70	8BL	Area 8B Lateral	Yes*	19,22	8B	Petredes and Pandya 1999	https://www
71	9p	Area 9 Posterior	Yes*	19,22	9	Petredes and Pandya 1999	https://www
72	10d	Area 10d	Yes*	19,20,22	10, Fp1, Fp2	Petredes and Pandya 1999, Bludau et al 2014[55]	https://www https://www
73	8C	Area 8C	Yes*	8,21,22	8Av	Petredes and Pandya 1999	https://www
74	44	Area 44	No	8,12,21	44d, 44v	Fischl et al 2008, Amunts et al 1999, Amunts et al 2010	https://www https://www https://www
75	45	Area 45	No	12,21	45a, 45p	Fischl et al 2008, Amunts et al 1999, Amunts et al 2010	https://www https://www https://www
76	47l	Area 47l (47 lateral)	No	12,20,21		Van Essen et al 2012b, Ongur et al 2003	https://www https://www
77	a47r	Area anterior 47r	Yes*	20,21,22	47r	Van Essen et al 2012b, Ongur et al 2003	https://www https://www

78	6r	Rostral Area 6	No	8,9,12,21		Amunts et al 2010	https://ww
79	IFJa	Area IFJa	Yes	8,21,22			
80	IFJp	Area IFJp	Yes	8,21,22			
81	IFSp	Area IFSp	Yes	21,22			
82	IFSa	Area IFSa	Yes	21,22			
83	p9-46v	Area posterior!9-46v	Yes*	21,22	9-46v	Petredes and Pandya 1999	https://ww
84	46	Area 46	No	21,22		Petredes and Pandya 1999, Rajkowska and Goldman-Rakic 1995a[103], Rajkowska and Goldman-Rakic 1995b[104]	https://ww https://ww https://ww
85	a9-46v	Area anterior!9-46v	Yes*	20,21,22	9-46v	Petredes and Pandya 1999	https://ww
86	9-46d	Area 9-46d	No	20,22		Petredes and Pandya 1999	https://ww
87	9a	Area 9 anterior	Yes*	19,20,22	9	Petredes and Pandya 1999	https://ww
88	10v	Area 10v	Yes	19,20	10, Fp2	Bludau et al 2014	https://ww
89	a10p	Area anterior 10p	Yes*	20,22	10p, 10, Fp1	Van Essen et al 2012b, Ongur et al 2003, Bludau et al 2014	https://ww https://ww https://ww

90	10pp	Polar 10p	Yes*	19,20	10p, 10, Fp1	Van Essen et al 2012b, Ongur et al 2003, Bludau et al 2014	https://www https://www https://www
91	11l	Area 11l	No	20	Fo3	Van Essen et al 2012b, Ongur et al 2003, Henssen et al 2016[80]	https://www https://www https://www
92	13l	Area 13l	No	20	Fo3	Van Essen et al 2012b, Ongur et al 2003, Henssen et al 2016	https://www https://www https://www
93	OFC	Orbital Frontal-Complex	Yes*	19,20	11m, 13b, 13m, 14r,Fo1	Van Essen et al 2012b, Ongur et al 2003, Henssen et al 2016	https://www https://www https://www
94	47s	Area 47s	No	12,20		Van Essen et al 2012b, Ongur et al 2003	https://www https://www
95	LIPd	Area LateralIntraParietal dorsal	Yes*	16,17		Van Essen et al 2012a	https://www
96	6a	Area 6 anterior	Yes	7,8,22	6, 6a_	Fischl et al 2008, Geyer 2004, Geyer et al 2000	https://www http://www https://www
97	i6-8	Inferior 6-8Transitional Area	Yes*	8,22	FC(B)	von Economo and Koskinas 1925, Triarhou 2007[116]	https://www https://www

98	s6-8	Superior 6-8Transitional Area	Yes*	7,8,22	FC(B)	von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
99	43	Area 43	No	6,8,9,12	41	Brodmann 1909[56], Brodmann 2007[52], Nieuwenhuys et al 2014	https://www https://www https://www
100	OP4	Area OP4/PV	No	6,9,17	68	Eickhoff et al 2006a[63], Eickhoff et al 2006b[62],Nieuwenhuys et al 2014	https://www https://www https://www
101	OP1	Area OP1/SII	No	9,10		Eickhoff et al 2006a, Eickhoff et al 2006b	https://www https://www
102	OP2-3	Area OP2-3/VS	Yes*	9,10,12	OP2,OP3	Eickhoff et al 2006a, Eickhoff et al 2006b	https://www https://www
103	52	Area 52	No	10,12	IBT	Brodmann 1909, Brodmann 2007, von Economo and Koskinas 1925, Triarhou 2007	https://www https://www https://www https://www
104	RI	RetroInsular Cortex	No	9,10,12,15	reI, reIt,RetroInsular, Belt,TD	Glasser and Van Essen 2011, Pandya and Sanides 1973[99], Kurth et al 2009[87], von Economo and Koskinas 1925, Triarhou 2007	https://www https://www https://www https://www https://www

105	PFcm	Area PFcm	No	9,10,15,17		Caspers et al 2006[59], Caspers et al 2008[58]	https://www https://www
106	PoI2	Posterior Insu-larArea 2	Yes*	12	Id1, Id2, Id3	Kurth et al 2009, Morel et al 2013	https://www https://www
107	TA2	Area TA2	No	10,11,12	TE1.2	von Economo and Koski-nas 1925, Triarhou 2007,Morosan et al 2001[93]	https://www https://www https://www
108	FOP4	Frontal OPercu-larArea 4	Yes	9,12,21			
109	MI	Middle In-sular Area	Yes*	12	Ial	Van Essen et al 2012b, Ongur et al 2003	https://www https://www
110	Pir	Pirform Cortex	No	12,14,20	Poc	Glasser and Van Essen 2011, Ding et al 2009[61], Morel et al 2013[92]	https://www https://www https://www
111	AVI	Anterior VentralIn-sular Area	Yes*	12,20,21	Iai	Van Essen et al 2012b, Ongur et al 2003	https://www https://www

112	AAIC	Anterior Agranu-larInsula Complex	Yes*	12,20	Iai, Ial	Van Essen et al 2012b, Ongur et al 2003	https://www https://www
113	FOP1	Frontal OPercu-larArea 1	Yes	8,9,12			
114	FOP3	Frontal OPercu-larArea 3	Yes	12			
115	FOP2	Frontal OPercu-larArea 2	Yes	9,12			
116	PFt	Area PFt	No	6,16,17		Caspers et al 2006, Caspers et al 2008	https://www https://www
117	AIP	Anterior IntraPari-etalArea	Yes*	6,16,17		Van Essen et al 2012a	https://www
118	EC	Entorhinal Cortex	No	13	28	Fischl et al 2009	https://www

119	PreS	PreSubiculum	No	2,13,18	Sub, Subicular	Glasser and Van Essen 2011, Amunts et al 2005[50]	https://www https://www
120	H	Hippocampus	No	13			
121	ProS	ProStriate Area	No	1,2,13,18		Glasser and Van Essen 2011, Vogt et al 2001[122], Sanides and Vitzthum 1965[106], Sanides 1970	https://www https://www https://www http://www brain/oclc/6
122	PeEc	Perirhinal Ectorhinal-Cortex	Yes*	13,14	ATFP, AFP1,Ectorhinal,Perirhinal, 35, 36	Augustinack et al 2013[54], Ding et al 2009[61], Ding and Van Hoesen 2010, Rajimehr et al 2009, Tsao et al 2008[118]	https://www https://www https://www https://www https://www
123	STGa	Area STGa	Yes	11,12,14			
124	PBelt	ParaBelt Complex	Yes*	10,11	ParaBelt, TA1	Moerel et al 2014[91], von Economo and Koskinas 1925,Triarhou 2007	https://www https://www https://www
125	A5	Auditory 5 Complex	Yes	11,15			
126	PHA1	ParaHippocampal Area 1	Yes	2,4,13			
127	PHA3	ParaHippocampal Area 3	Yes	4,13,14			

128	STSda	Area STSd anterior	Yes	11,14			
129	STSdp	Area STSd posterior	Yes	11,15			
130	STSvp	Area STSv posterior	Yes	11,14,15			
131	TGd	Area TG dorsal	Yes*	11,12,13,14	TG	Ding et al 2009, von Economo and Koskinas 1925,Triarhou 2007	https://www https://www https://www
132	TE1a	Area TE1 anterior	Yes*	11,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
133	TE1p	Area TE1 posterior	Yes*	5,11,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
134	TE2a	Area TE2 anterior	Yes*	14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
135	TF	Area TF	No	4,13,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
136	TE2p	Area TE2 posterior	Yes*	4,5,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
137	PHT	Area PHT	No	5,11,14,15		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
138	PH	Area PH	No	4,5,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www
139	TPOJ1	AreaTemporoParietoOccipital Junction 1	Yes	10,14,15,17			

140	TPOJ2	AreaTemporoParietoOccipital Junction 2	Yes	5,14,15,17				
141	TPOJ3	AreaTemporoParietoOccipital Junction 3	Yes	5,15,17				
142	DVT	Dorsal Transition-alVisual Area	Yes	2,3,16,18				
143	PGp	Area PGp	No	5,15,17	39,PG	Caspers et al 2006, Caspers et al 2008	https://www https://www	
144	IP2	Area Intra-Parietal 2	No	16,17		Choi et al 2006[60]	https://www	
145	IP1	Area Intra-Parietal 1	No	16,17		Choi et al 2006	https://www	
146	IP0	Area Intra-Parietal 0	Yes	3,5,16,17				
147	PFop	Area PF opercular	No	6,9,17	40, 72	Caspers et al 2006, Caspers et al 2008, Nieuwenhuys et al 2014	https://www https://www https://www	
148	PF	Area PF Complex	No	9,15,17	40, 88	Caspers et al 2006, Caspers et al 2008, Nieuwenhuys et al 2014	https://www https://www https://www	
149	PFm	Area PFm Complex	No	15,17	40, 89	Caspers et al 2006, Caspers et al 2008, Nieuwenhuys et al 2014	https://www https://www https://www	

150	PGi	Area PGi	No	15,17	PGa, 39, PG, 90	Caspers et al 2006, Caspers et al 2008, Nieuwenhuys et al 2014	https://www https://www https://www
151	PGs	Area PGs	No	15,17	PGa, 39, PG, 90	Caspers et al 2006, Caspers et al 2008, Nieuwenhuys et al 2014	https://www https://www
152	V6A	Area V6A	No	3,18	112	Pitzalis et al 2013, Nieuwenhuys et al 2014	https://www https://www
153	VMV1	VentroMedial VisualArea 1	Yes*	2,4,13	PHC2, PHC-2	Arcaro et al 2009[53], Wang et al 2015[126]	https://www https://www
154	VMV3	VentroMedial VisualArea 3	Yes*	2,4,13	VO2	Arcaro et al 2009, Wang et al 2015, Wandell and Winawer 2011	https://www https://www https://www
155	PHA2	ParaHippocampalArea 2	Yes	4,13			
156	V4t	Area V4t	No	5	LO2	Abdollahi et al 2014, Kolster et al 2010, Larsson and Heeger 2006	https://www https://www https://www
157	FST	Area FST	No	5,14,15		Abdollahi et al 2014, Kolster et al 2010	https://www https://www
158	V3CD	Area V3CD	Yes	2,3,5,17	V3A,V3B, hOC4la	Abdollahi et al 2014, Malikovic et al 2015	https://www https://www

159	LO3	Area Lateral Occipital3	Yes	5,15,17	hOC4la			
160	VMV2	VentroMedial VisualArea 2	Yes*	2,4,13	PHC1, PHC-1	Arcaro et al 2009, Wang et al 2015	https://www https://www	
161	31pd	Area 31pd	Yes*	18	31, 31d, 31v	Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www	
162	31a	Area 31a	Yes*	18	31, 31d, 31v	Vogt 2009, Palomero-Gallagher et al 2009	https://glob neurobiology and- disease- 97801985669 https://www	
163	VVC	Ventral Visual-Complex	Yes*	4,13,14	VO1, VO2, PHC1,PHC2, PHC-1, PHC-2, FG1	Arcaro et al 2009, Wang et al 2015, Wandell and Winawer 2011, Caspers et al 2013, Weiner et al 2014	https://www https://www https://www https://www https://www	

164	25	Area 25	No	19,20		Vogt 2009, Palomero-Gallagher et al 2015	https://glob neurobiology and- disease- 97801985669 https://www
165	s32	Area s32	No	19	32pl, 32	Vogt 2009, Palomero-Gallagher et al 2015, Van Essen et al 2012b, Ongur et al 2003	https://glob neurobiology and- disease- 97801985669 https://www https://www https://www
166	pOFC	posterior OFCCom-plex	Yes*	12,19,20	13a, 14c, Fo2	Van Essen et al 2012b, Ongur et al 2003, Henssen et al 2016	https://www https://www https://www
167	PoI1	Area Posteri-orInsular 1	Yes*	12	Id1, Id2, Id3	Kurth et al 2009, Morel et al 2013	https://www https://www
168	Ig	Insular Granular-Complex	Yes*	9,12	Ig1, Ig2	Kurth et al 2009, Morel et al 2013	https://www https://www

169	FOP5	Area FrontalOp-ercular 5	Yes	12,21	PrCO	Glasser and Van Essen 2011	https://www
170	p10p	Area posterior 10p	Yes*	20,22	10p, 10, Fp1	Van Essen et al 2012b, Ongur et al 2003, Bludau et al 2014	https://www https://www https://www
171	p47r	Area posterior 47r	Yes*	20,21,22	47r	Van Essen et al 2012b, Ongur et al 2003	https://www https://www
172	TGv	Area TG Ventral	Yes*	13,14		Ding et al 2009, von Economo and Koskinas 1925, Triarhou 2007	https://www https://www https://www
173	MBelt	Medial Belt Complex	Yes*	10,12	Belt, TB	Moerel et al 2014, von Economo and Koskinas 1925,Triarhou 2007	https://www https://www https://www
174	LBelt	Lateral Belt Complex	Yes*	10	Belt, TB	Moerel et al 2014, von Economo and Koskinas 1925,Triarhou 2007	https://www https://www https://www
175	A4	Auditory 4 Complex	Yes*	11,15	TE3	Morosan et al 2005	https://www
176	STSva	Area STSv anterior	Yes	11,14			
177	TE1m	Area TE1 Middle	Yes*	11,14		von Economo and Koskinas 1925, Triarhou 2007	https://www https://www

178	PI	Para-Insular Area	No	11,12,14	IBT	von Economo and Koskinas 1925, Triarhou 2007, Ding et al 2009	https://www https://www https://www
179	a32pr	Area anterior 32prime	Yes*	19	32’	Vogt 2009	https://glob neurobiology-and-disease-97801985669
180	p24	Area posterior 24	Yes*	19	24	Vogt 2009	https://glob neurobiology-and-disease-97801985669

C Survey

Q: How useful is the CE from a Neuroscience/Research perspective? (1 - 5)
A: 4, 3, 3, 3, 5

Q: Notes/Reasons?
A: Not enough information provided neuroscience-wise. ”Sections” not explained (small information icons could be useful). Studies could be put on a list.

Q: How informative is the CE to members of the public?

A: 3 4 4 5 3

Q: Notes/Reasons?

A: It would be helpful for public to have more info on the function of regions. Descriptions could be enriched by adding images (e.g. auditory area showing an illustration of the auditory system)

Q: How useful could CE be as a teaching tool?

A: 4 4 4 5 5

Q: Notes/Reasons?

A: other notes: failed on chrome on macbook pro (no webgl error). worked on Safari. unclear how to get back to original view. I think this is a very nice start on a piece of software that displays our brain map. As currently implemented I think it would mainly be helpful for teaching or public exploration, however, with additional features it might become very useful for research as well.

Q: Are the set of standard views (e.g front, left, top) along with the orbit controls (e.g zoom, rotate, pan), and the ability to hide each hemisphere of the brain, sufficient to navigate the brain effectively?

A: Yes. yes Yes. Some information about these controls should be provided somewhere. Yes for now very efficient

Q: Is the information presented for each parcel clear and readable?

A:The information is readable, but not always clear (e.g. what does the * mean when an area is new). See notes in the previous page. Yes yes

Q:Is there any other information you would like to see about each parcel?

A: Maybe brief introduction to the area's function would be nice. Also there is no information for some areas, e.g. Left STSva. See notes in the previous page. Yes, task activity

Q: With Labels turned on, are they readable/ a good size?

A: The labels have a good size but appear blurry in some locations. yes No. Yes though I might customize a bit reasonably visible

Q: Would the ability to move labels around manually to reduce overlap be useful?

A:Yes, especially for presentation purposes. not really It might make things messy. Rotation seems to be enough not sure

Q: The brain shown is an 'inflated' model, would viewing different levels of inflation (e.g pial, flat and very inflated) be useful?

A: Not necessarily, mostly useful for screenshots. not really Yes. Yes not sure

Q: Are the parcels a good thickness? Would using cortical thickness data to scale the models be a useful upgrade on the model?

A: Parcels are a good thickness, using cortical thickness instead would be nice but not essential. not sure. seems good. Yes. I think they are a bit too thick and using cortical thickness might be a good way to fix this yes

Q: In the 'More' section you can find some of the types of data used to generate this parcellation (e.g myelin), are these useful? how could this extra information be improved, would viewing fiber tracts be useful?
A: It would be useful to show colorbars for the different types of data. And potentially visualise this information along with the parcels. At the moment this does not seem to be possible. Also when an option from the 'More' section is selected, the UI is not responsive anymore (e.g. the panel showing the parcel information cannot be hidden anymore). yes & yes. Fibers, definitely. I would also like to upload my own data if given the functionality... Overlaying parcels to underlying data (other parcellation, myelin maps, etc) could also be cool. Yes, I don't know about fiber tracts, would require more discussion... yes

Q: Are the controls on the screens clear? is it intuitive what they are for?
A: The controls on the screen are clear. It gets a bit confusing when viewing one of the 'More' maps, since 'LEFT', 'RIGHT', 'LABELS' and the colorbar are still visible but cannot be used for interaction with the volume. mostly. Yes. Explode is cool, but not tells much until you click. Yes in general (connectivity controls were not obvious to me) relatively clear but could not reset back to parcels once I visualised myelin

Q: Should the controls always be on the screen or in a hideable menu?
A: I think a hideable menu would be preferable so that the user can choose

which controls to view according to their convenience. seems good as is I would like to see a "full-screen" mode where I can only interact with the brain. Always on screen with option to hide I suppose. yes

Q: Should the background be a light or a dark color?
A: Toggle between based on preference. Toggle between based on preference. Toggle between based on preference. Light

Q: What features are the strongest and why?
A: The explode feature and the information provided for each region (along with valid URLs to the original study findinds). This is a very intuitive way of exposing users to neuroscientific findings and extremely attractive for teaching purposes. Fast response time when clicked on a parcel or rotate/-pan. Being able to display HCP data interactively on the web 3D interactive visualisation

Q: Which features are the weakest and why?
A: The introduction of the additional maps (myelin, correlation etc.) seems still immature. They need to be accompanied with colorbars and it would also be useful to have a label indicating which kind of modality is currently viewed. There are also controls visible that do not interact with the volume. bit slow. Labels (not easy to read, look messy), pop-up parcels (instead they could be simply highlighted when selected) The tool could be expanded quite a bit I think. Also the explode is about 3x too far. not sure

Q: What else would you like to see added to the CE? e.g links to the papers about each parcel, uploading your own data

A: Uploading data in a standard format would be very useful. Long-term features would include visualisation of the tracts and potentially the connectivity networks. for public engagement, mainly general neuroscience info Uploading your own data, overlaying parcels on top of other maps. I think the single biggest potential improvement would be integration with the BALSA database, which does not currently have an interactive viewer, but could use one. search for specific regions

Q: Any other feedback?

A: Great work, nice features already implemented and more of interest can easily be integrated. great! Rendering took couple of minutes when I opened the page on a weak connection. Fans go crazy on a 2013 MacBook using Chrome. Thanks for working on this. Really impressive! –Matt Glasser.