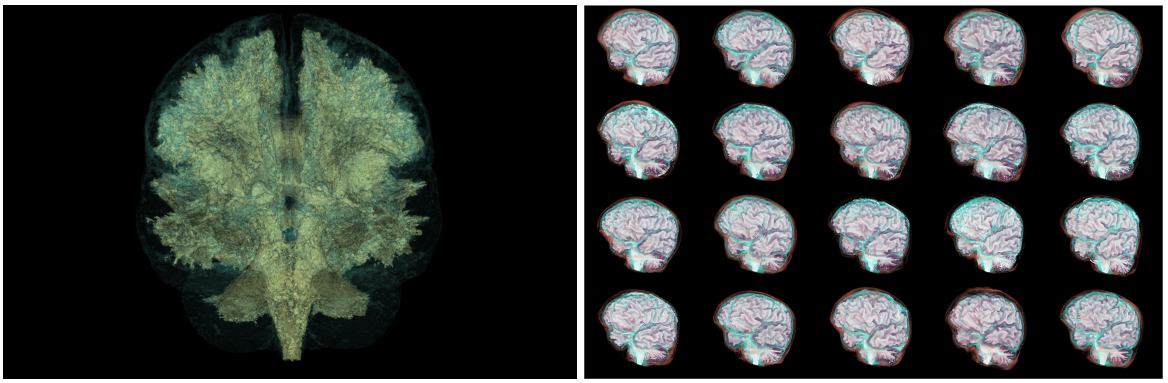
Comparative Visualization of Brain Anatomy Variability

Brunner Lukas 11909464
Prohaska Jonas 01449302



Left: white matter where volumes of 10 out of 20 subjects agree at any given point (with slight context of gray matter; *Right:* interactive juxtaposition of 20 volumes in a combination of cerebrospinal fluid (cyan), bone marrow (orange), white matter (white) and slight gray matter (gray)

Motivation

The project focussed on visualizing differences in brain anatomy using some of the fundamental approaches for series of spatial 3D data sets identified by Kim et al. [2017]. We stuck to the more fundamental approaches such as interactively configurable juxtaposition, interchangeable visualizations with animated transitions, superimposition and explicit encoding. The user is invited to play with the settings and explore the data using all available interactions.
We did not choose to implement the suggested YMCA algorithm [J. Schmidt et al. 2014] since after some exploration of the data it turned out that the interesting parts of the brain either highly vary across patients (especially White Matter and also Gray Matter) or mainly contain very fine details (blood vessels, dura mater/membranes) that may exist in disjoint places across patients even if the data is in average registered to each other. Thus, we focussed on visualizations that show where data sets are similar and where they diverge.

Tech Stack

As our technology stack we chose Python with VTK and Qt. We expected VTK to allow rapid prototyping and Qt to offer an easy way to integrate state-of-the-art UI into the project. Both assumptions were true, however it still took quite some effort to make things work for the user in a relatively painless way. This was due to requirements such as custom input handling that synchronizes multiple render windows, VTK off-screen rendering from those render windows while hidden in Qt and still respecting window resizing or dealing with 3 different reference-counting systems where one of them (VTK) may produce cyclic references and thus memory leaks if not disassembled correctly.
Although we did our best in the available time to make the program usable. However, for the

best experience we recommend using a machine with a decent GPU (e.g. 8-10GB video memory), CPU and RAM (e.g. >=8 GB).

How To Use

Installation

The project is built and tested using Python 3.9.7. Python versions below 3.8 are guaranteed to not work. The library dependencies are `vtk`, `pyside6` and `numpy`. We recommend creating a virtual environment using the `requirements.txt` file. This can be done manually or with the provided `install.bat` file. Under Windows it assumes Python to be installed with pip and available from the command line via the `python` command. Under Linux it assumes python to be available as `python3.9`. To run the application from the virtual environment open a command prompt/shell next to the `install.bat` file and execute:

On Windows:	On Linux:

`call install.bat`
`pushd code`
`python Main.py`
	`source install.bat`
	`source ./venv/bin/activate`
	`pushd code`
	`python3.9 Main.py`

On Linux make sure that `install.bat` has Linux file endings and the application access to a properly set up windowing system. Generally we recommend using Windows for a smoother installation experience.
When first starting the program it loads all `.mnc` files it finds in the `../Data` folder relative to the working directory. The data files come with the package zipped in `./Data/datafiles.zip` and must first be extracted to the Data folder.

Features

The program reads all those .mnc files and per default displays the Gray Matter and Vessels labels of the first volume. Before going into the various sliders and buttons, the user is advised to change the amount of available GPU Memory unter Settings->Set GPU Memory Usage. If the program uses up its set GPU memory quota the rendering mode is automatically switched to CPU for subsequent volumes. The performance will not be great but it keeps the application from crashing inside VTK. We do not rely on platform-dependent tools such as nvidia-smi (that might or might not be installed on the host) to query the GPU memory load. Hence, we task the user with administering this setting and figuring out what works. Note: typically CPU renderers will also have a different appearance.

The UI is split into a Selectable Volume List and an Interactive Render Module where various comparative visualizations can be chosen from and adjusted. The Volume List explains itself: those volumes that are selected will be used for the comparative visualizations on the right. The starting view is a juxtaposition. So adding volumes will add render windows next to already existing renders. All render windows can be interacted with like any other VTK renderer, so we advise you refer to https://vtk.org/doc/nightly/html/classvtkInteractorStyle.html#details to see all available options. To the left of the rendering window(s) are sliders that define the transfer function for

all labels of the brain data, from left (completely transparent) to right (completely opaque). The user can also change the color of any part by clicking on the label (it is a button with a slider inside). Towards the top, the “Shaded” button changes the VTKRenderer to add lighting calculations and shadows (for best results in with shaded mode, set labels to more bright and opaque colors), “Progressive Rendering” is activated per default, turning it off turns off adaptive sampling, it will be slower but always display sharp images. When running mostly on integrated graphics or CPU this can yield better images at the expense of interactiveness.

To the right of Progressive Rendering is also the button that enables the Interchangeable View, which blends through all selected volumes and can even animate said blending from the first selected volume to the last. Please note that doing this for all 20 Volumes can be extremely demanding (depending on the screen size of the image), since it always blends all volumes when changing the slider, instead of just the previous and next volume. This is a limitation arising from the direct use of vtkImageBlend.

The third comparative visualization is found in the Explicit Encoding tab at the top of the Interactive Render Module. The transfer functions behave the same as in the Juxtaposition View but separate settings can be applied. Additionally to selecting volumes it is now possible to also select the labels that should be visualized using Explicit Encoding. In Explicit Encoding, each selected label can be combined across all selected volumes via a dropdown at the top: Union and Intersection correspond to the voxel-wise logical OR and AND operations across volumes.
The third option is the Addition, where the user can use the slider to set a threshold (e.g. n=2). Then only those voxels of the labels are segmented and rendered where at least n (2 in our example) volumes of any selected volumes have their respective voxel inside the respective label. This differs from the logical AND operator in that it relaxes its constraints and allows all possible combinations of n data sets to contribute to the boolean inclusion into the segmented volume. It is implemented as the addition of the boolean label masks (1 where the label is, 0 everywhere else) and computing an isosurface at the given value for n. Examples are shown in the annex. Note: it may seem like the slider may not work at first, but even if the handle stays at the previous location, still drag it (blindly) to the final desired position and the handle will move after the update is completed.

Statement (Who did what)

Name	**Task**
Brunner/Prohaska	Setup VTK Renderer in QT
Prohaska	MINC Loading/Transform to and from Numpy
Brunner	Selectable Volume List
Prohaska	Add multiple Renders in Juxtaposition
Brunner	Synchronize User Interaction on all Renders

Prohaska	Set Transfer Functions via Slider and Color Picker
Brunner	Interchangeable View & Animation
Brunner	Explicit Encoding
Prohaska	UI Styling
Brunner/Prohaska	Testing/Optimization/Bug Fixing
Brunner/Prohaska	Report / Presentation

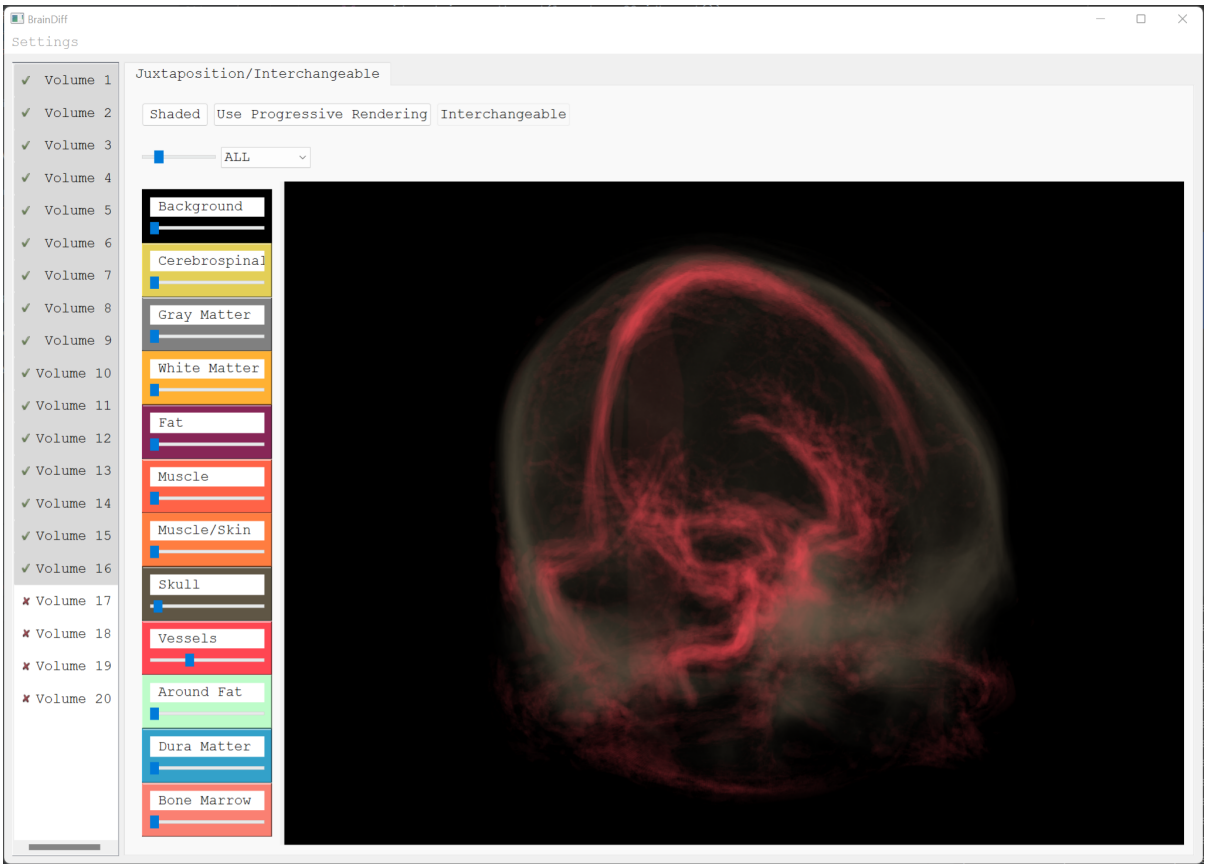
References

K. Kim et al., “Comparison techniques utilized in spatial 3D and 4D data visualizations: A survey and future directions”, Computers & Graphics (2017)

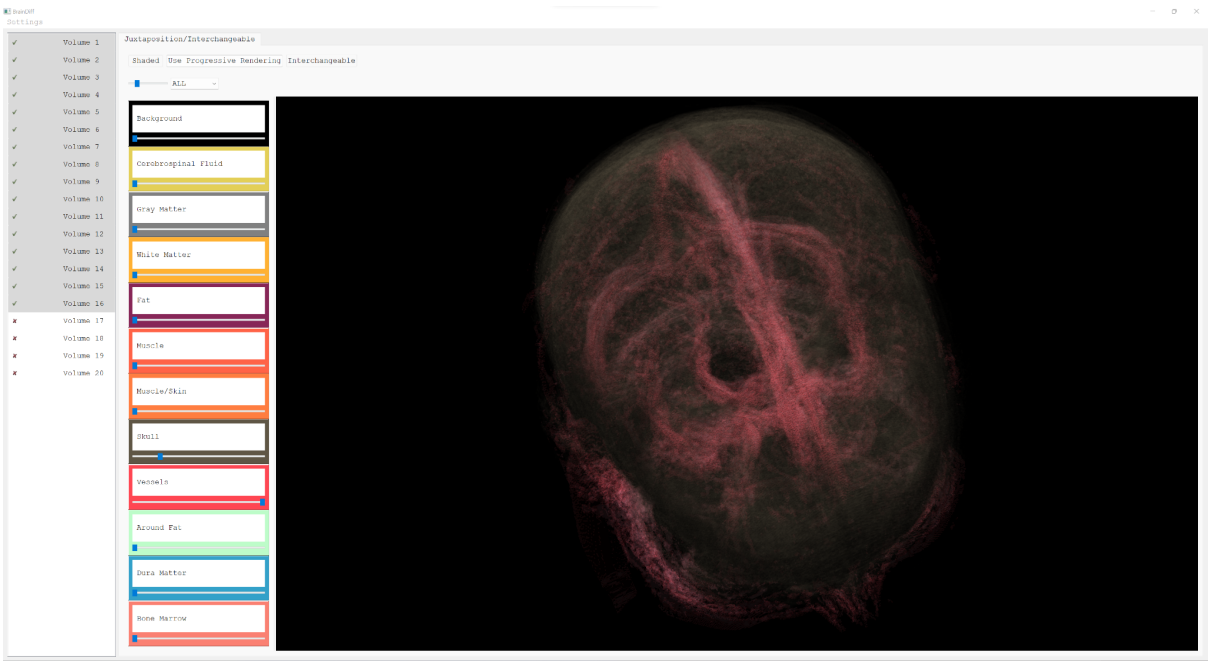
Schmidt, Johanna et al., “YMCA — Your mesh comparison application.”, 2014 IEEE Conference on Visual Analytics Science and Technology (VAST) (2014): 153-162.

Appendix: Screenshots

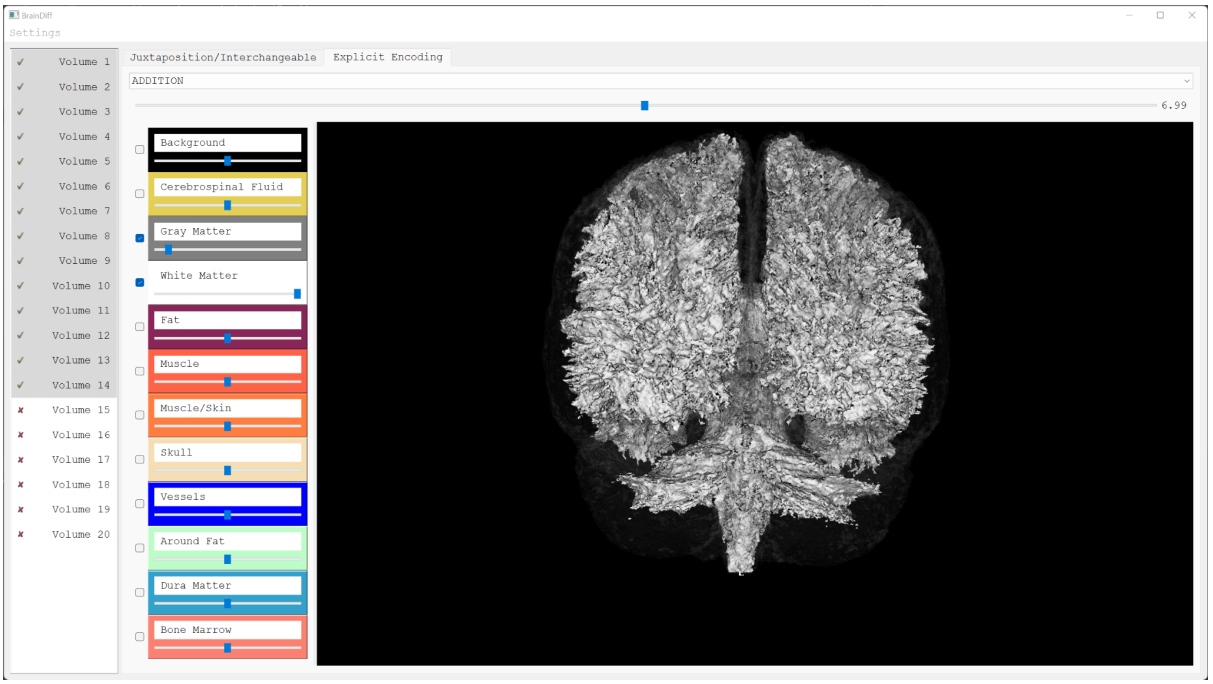
Overlays provide an intuitive overview of location and uncertainty:



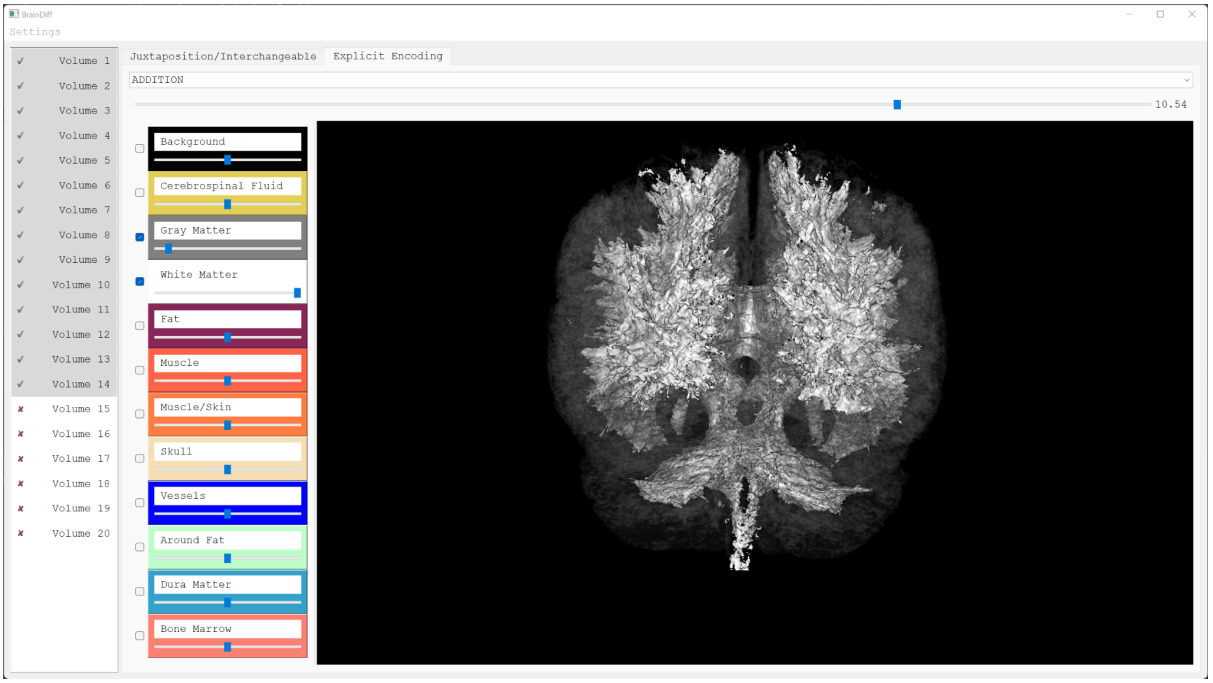
Overlays of blood vessels with a slight context of the skull.
Top: unshaded;
Bottom: shaded mode provides a bit more depth cues at the expense of slightly increased occlusion



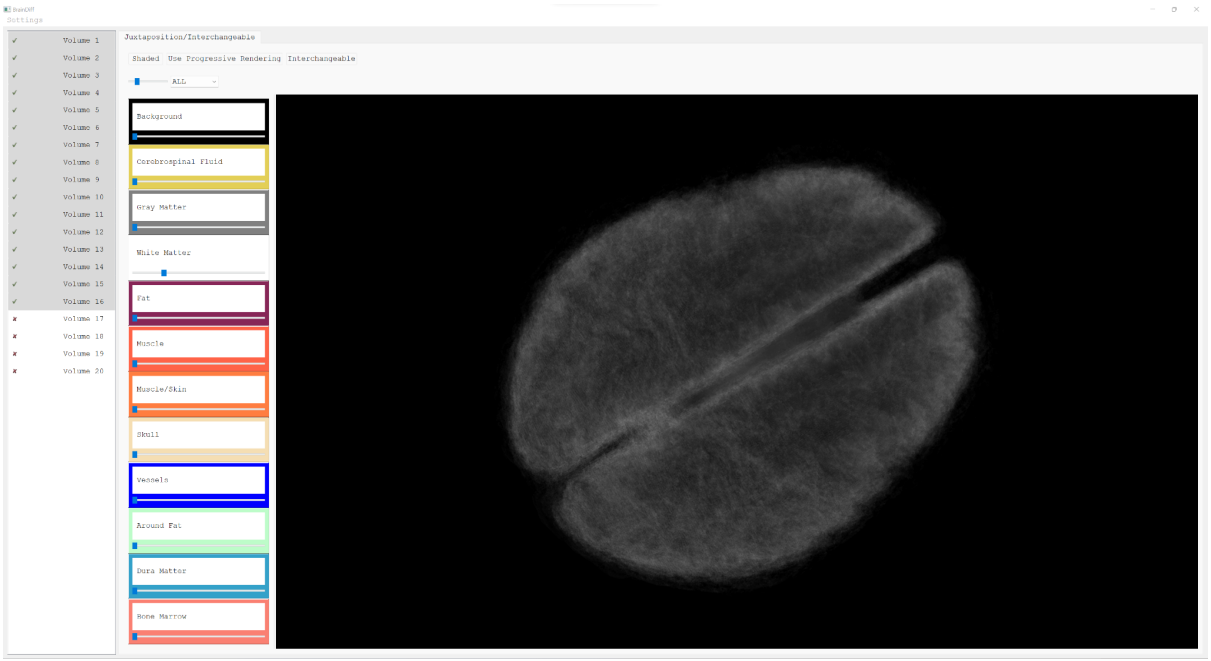
Addition operator in Explicit Encoding view. Provides a good overview on where data sets agree given a certain “confidence value”:



Top: the inside of the solid is covered by White Matter by at least 50% of 14 patients at any contained voxel. The contributing patients do not have to be the same across the underlying voxels, but the number of patients satisfies the limit set by the slider;
Below: at least any 75% of patients cover the visualized solid area;
Both images show a slight context of Gray Matter

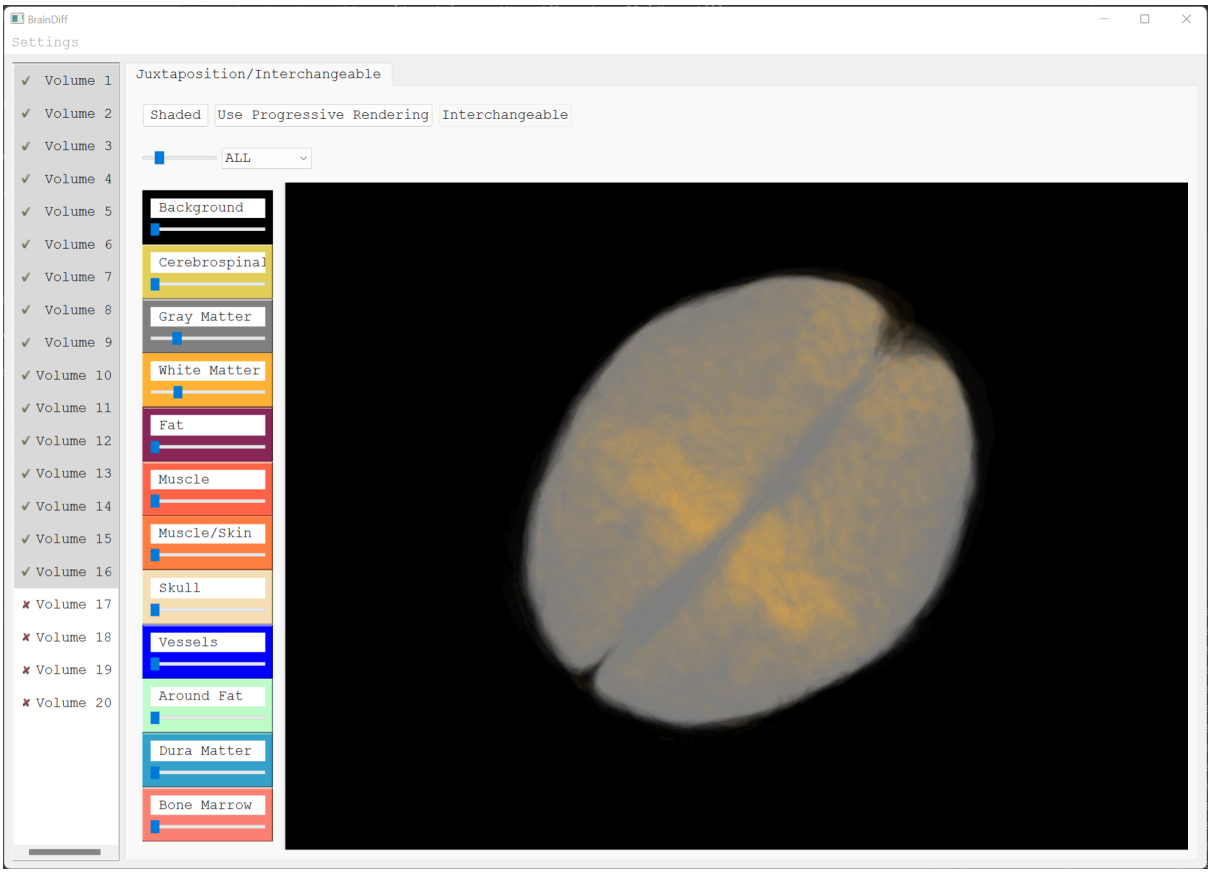


Shaded Overlay:



Overlay of shaded White Matter of 16 patients. White Matter variability is pretty high as shown by the previous visualization. The maximum possible location where to find it is (obviously) contained by the Dura Mater and less likely to extend very close to the hard boundary but else it may penetrate Mray Matter at almost any location. Thus the cloudy appearance fading out towards the boundary in the overlay. The shaded version conveys more information about the shape and structure of the boundary.

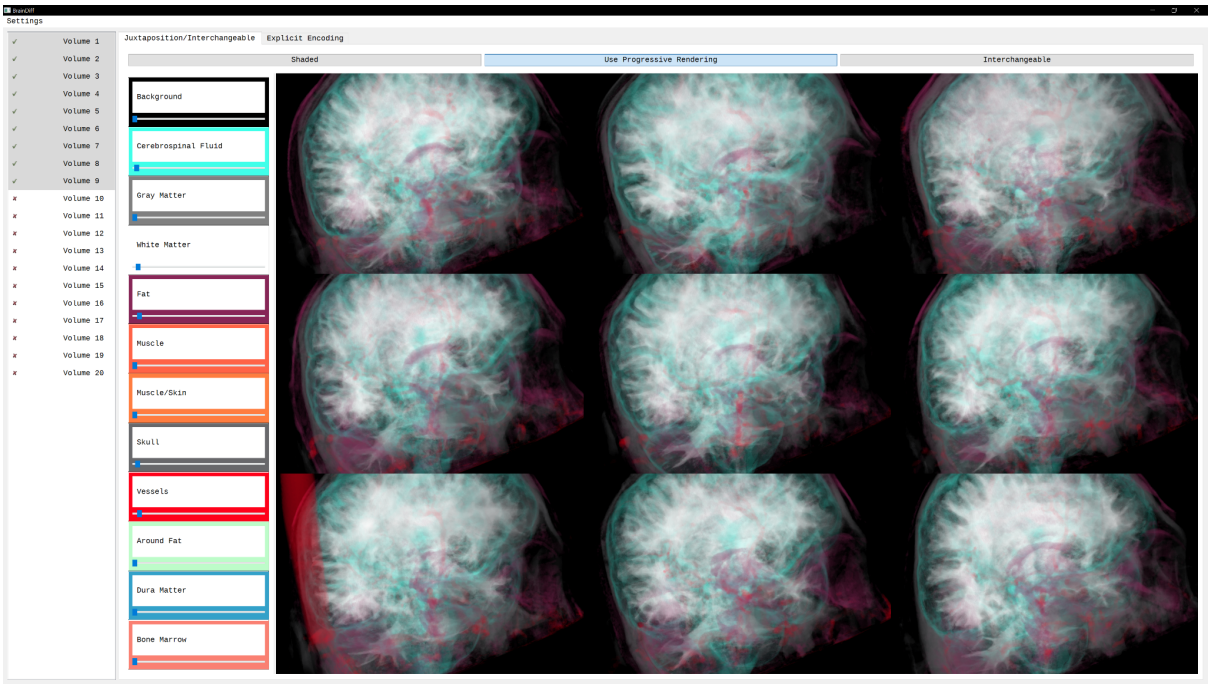
Non-shaded Overlay:

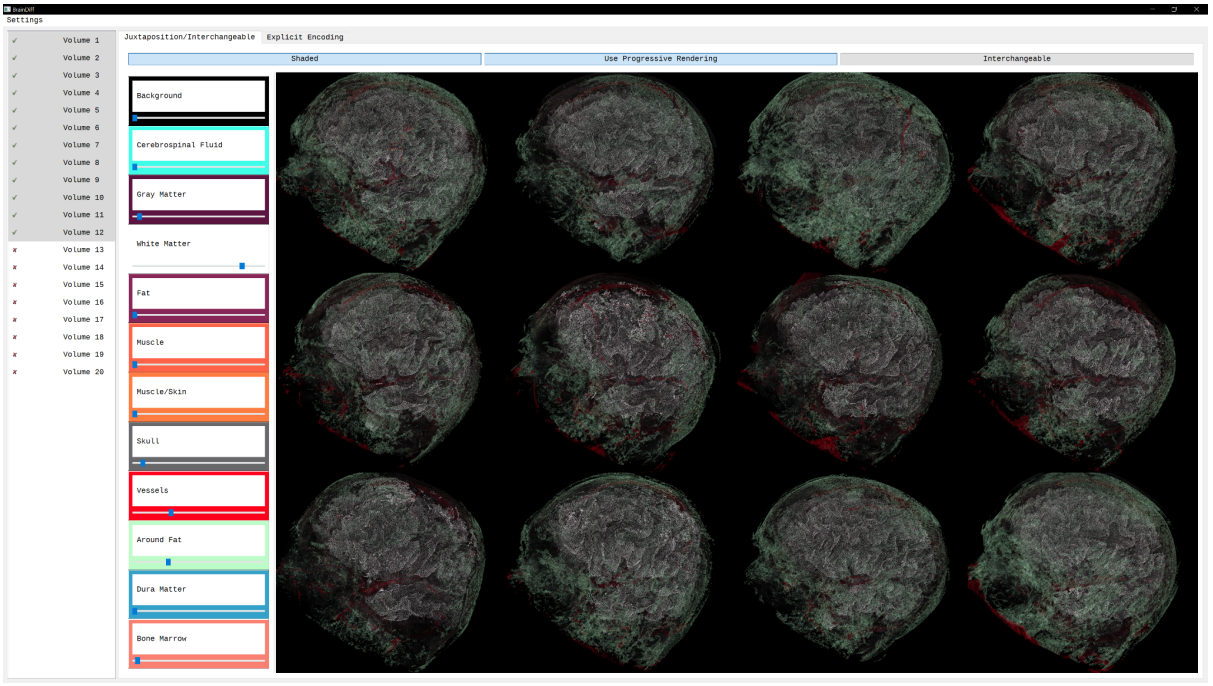


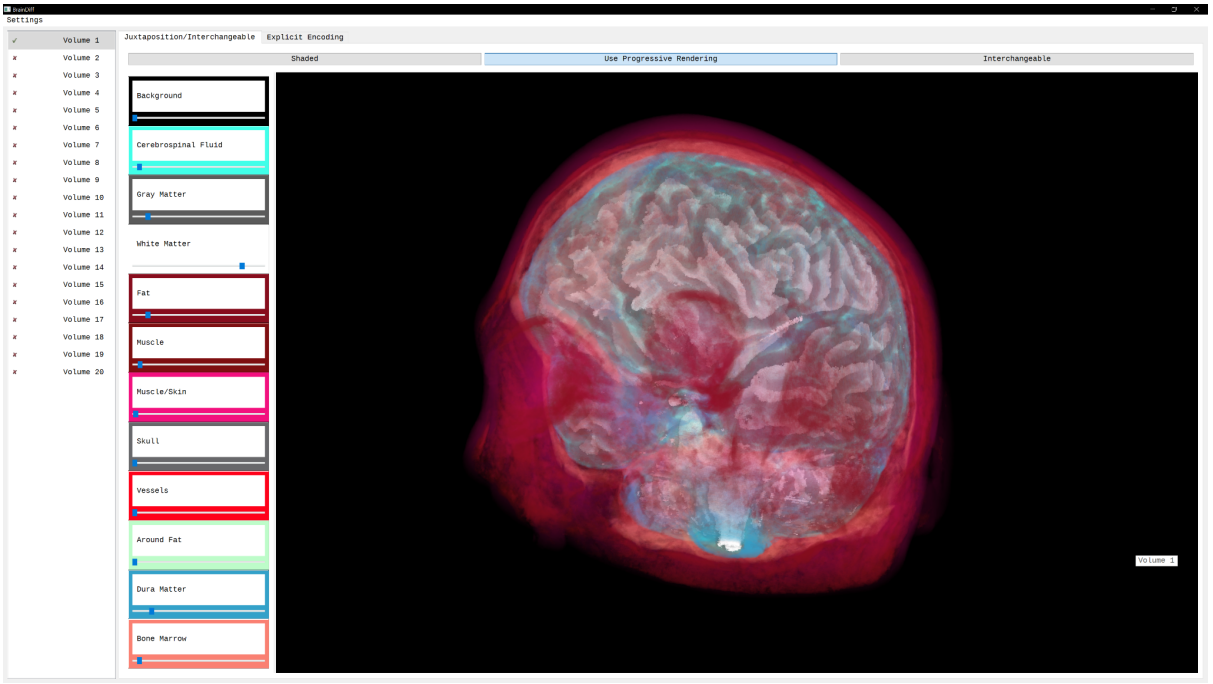
Overlay of 16 patients with transparent Gray (gray) and White Matter (orange). It seems that generally humans have a strip-like region where the White Matter reaches more close to the outer boundary of the Gray Matter where the Gray Matter seems less

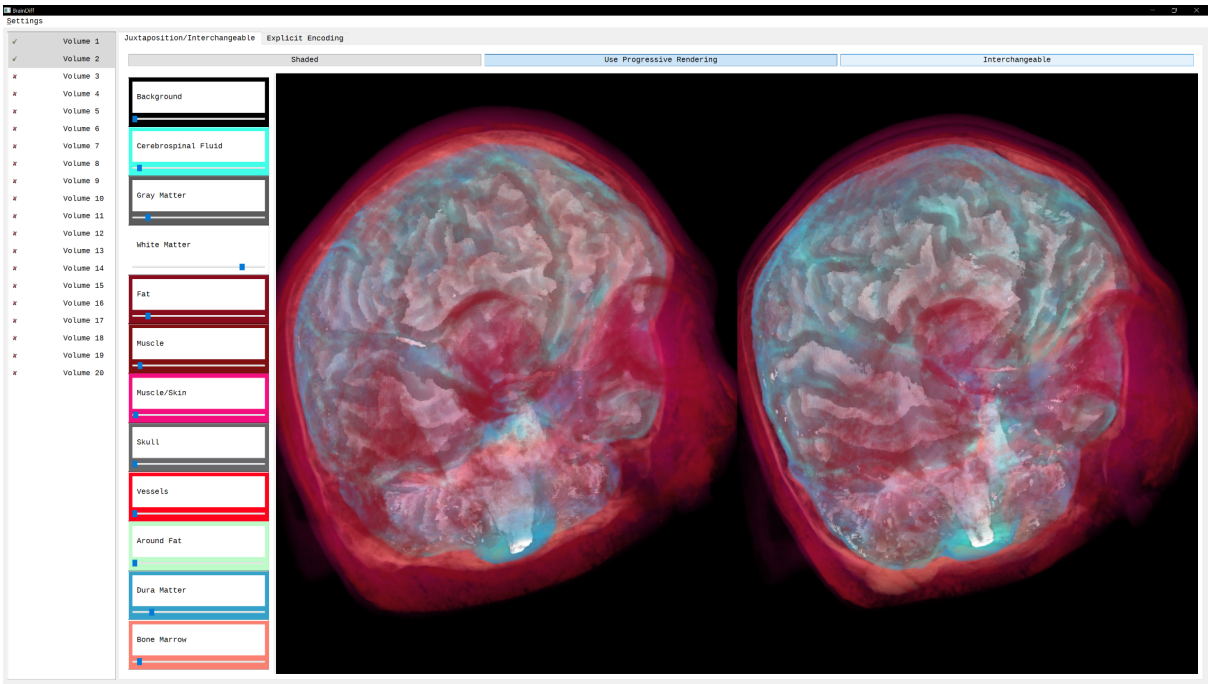
thick. The non-shaded version with transparency conveys more information on the depth relationship between labels inside of each other.

More screenshots:






BrainDiff
Settings
Volume 1
Volume 2
Volume 3
Volume 4
Volume 5
Volume 6
Volume 7
Volume 8
Volume 9
Volume 10
Volume 11
Volume 12
Volume 13
Volume 14
Volume 15
Volume 16
Volume 17
Volume 18
Volume 19
Volume 20
Juxtaposition/Interchangeable
Explicit Encoding
Shaded
Use Progressive Rendering
Interchangeable
Background
Cerebrospinal Fluid
Gray Matter
White Matter
Fat
Muscle
Muscle/Skin
Skull
Vessels
Around Fat
Dura Matter
Bone Marrow
Volume 1


BrainDiff
Settings
Volume 1
Volume 2
Volume 3
Volume 4
Volume 5
Volume 6
Volume 7
Volume 8
Volume 9
Volume 10
Volume 11
Volume 12
Volume 13
Volume 14
Volume 15
Volume 16
Volume 17
Volume 18
Volume 19
Volume 20
Juxtaposition/Interchangeable
Explicit Encoding
Shaded
Use Progressive Rendering
Interchangeable
Background
Cerebrospinal Fluid
Gray Matter
White Matter
Fat
Muscle
Muscle/Skin
Skull
Vessels
Around Fat
Dura Matter
Bone Marrow