MXB201 Project

Aiden Matthews, Jason Siu, Molly Cullen, Sofia Gabriela Nightingale – Group 29

GitHub - <https://github.com/mollyc9959/mxb_201_group26>

# Introduction

Within the medical field, imaging has become more and more important as the field progresses, with magnetic resonance imaging providing continuously insightful depictions of the brain and other soft tissue organs where x-rays provide lack-lustre information (Department of Health & Human Services, 2014). Although the images are helpful, are there additional ways to improve their insights? A possible way to enhance the information able to be gained is by using technology to estimate diffusion tensor imaging and how that may be used in combination with feature extraction. This report explores how technology may be used to achieve this, some of its possible drawbacks and overall methodology.

# Part I: MRI Diffusion Tensor Imaging

Diffusion-weighted MRI (magnetic resonance imaging) in the brain allows medical professionals to reconstruct the brain, in order to study brain anatomy and diagnose patients with potential conditions safely. Patients are exposed to a magnetic field, and the diffusivity of water in different locations of the brain tissue is measured. Through analysis of the properties of this diffusion, various figures can be produced to assist professionals with diagnosis.

## Estimating Diffusion Tensor from Raw Data

We are given a single slice of a scan for a patient, with the objective of using this data to estimate the diffusion tensor at each voxel.

From Jiang et al. (2005), we have the mathematical equation,

where *S* is the signal intensity, which decays exponentially as a function of the constant diffusion tensor  (mm2/s), the direction of the diffusion sensitising gradient pulse (a unit vector in ), and the parameter *b* (s/mm) the diffusion-weighting factor set by the machine operator. *b* is a scalar that absorbs all the details about the gradient pulse other than its direction, such as its strength and timing and is held constant for all the gradient pulses. The *b* value used for this analysis was 1000 s/mm (a typical value). It would be possible to change the value for alternative insights. Thus, the only independent variable changing throughout the scan are the directions **.**

We use this information, with the formula above to estimate , 3 x 3 symmetric positive definite matrix, at each voxel.

Substituting these values into Jiang’s equation above, we obtain a system of equations, where is the number of directions in . This can be written in matrix form. Since the initial equation is not linear, we will take the natural logarithm of each side, and construct a linear system of the form , where

, with and

Since we cannot find such that exactly (the system is inconsistent), we must find the most fitting solution for for which we can use the least squares method. The objective is to minimise the norm of the *residual*:

Solutions to the least squares problem:

can be found by solving the normal equations,

,

finding the QR decomposition,

then solving the triangular linear system using backward substitution.

For efficiency, we use MATLAB’s built-in Gram-Shmidt process (since we're using floating point arithmetic to output , from which we construct our 3x3 matrix .

## Results of Analysis

From our matrix D, we can obtain eigenvalues and their corresponding eigenvectors. D, being a symmetric positive definite matrix, has real and orthonormal eigenvectors. The three eigenvalues indicate the magnitude of diffusivity in three directions (Jiang et al., 2005).



*Figure 1 mean diffusivity*

Then, some common imaging techniques for diffusion tensor imaging are as follows:

### Mean diffusivity map

To determine the magnitude of the diffusion at each voxel, we find the mean diffusivity (mean average of all three eigenvalues) and produce the greyscale image to the right, simply by plotting the result, see figure 1. This map can help professionals monitor diffusion in the brain and signal potential areas of abnormal diffusion which may be of concern.

### Fractional anisotropy

To determine the fractional anisotropy, (a measure of how the eigenvalues differ), we use the formula:

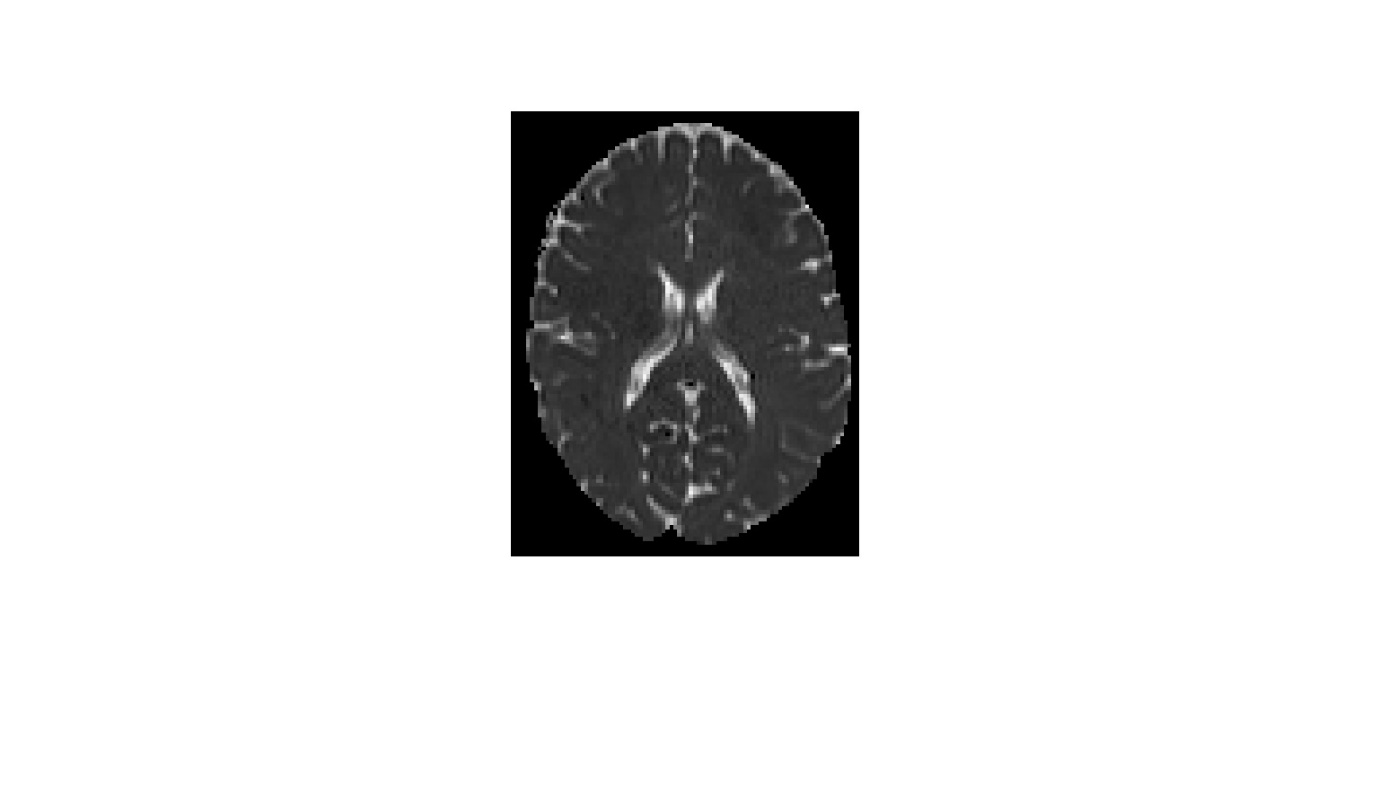


Figure 2 Fractional Anisotropy

(Elster 2009),

From our image, see figure 2, notice the very outside of the brain, and the x-shape inside, are much brighter. Thus, these areas are more anisotropic (Elster, 2009). Within these brighter sections the diffusion occurs very directionally instead of universally. Visualising this information can indicate various injuries or disease and assist in localizing them.

### Principal diffusion direction

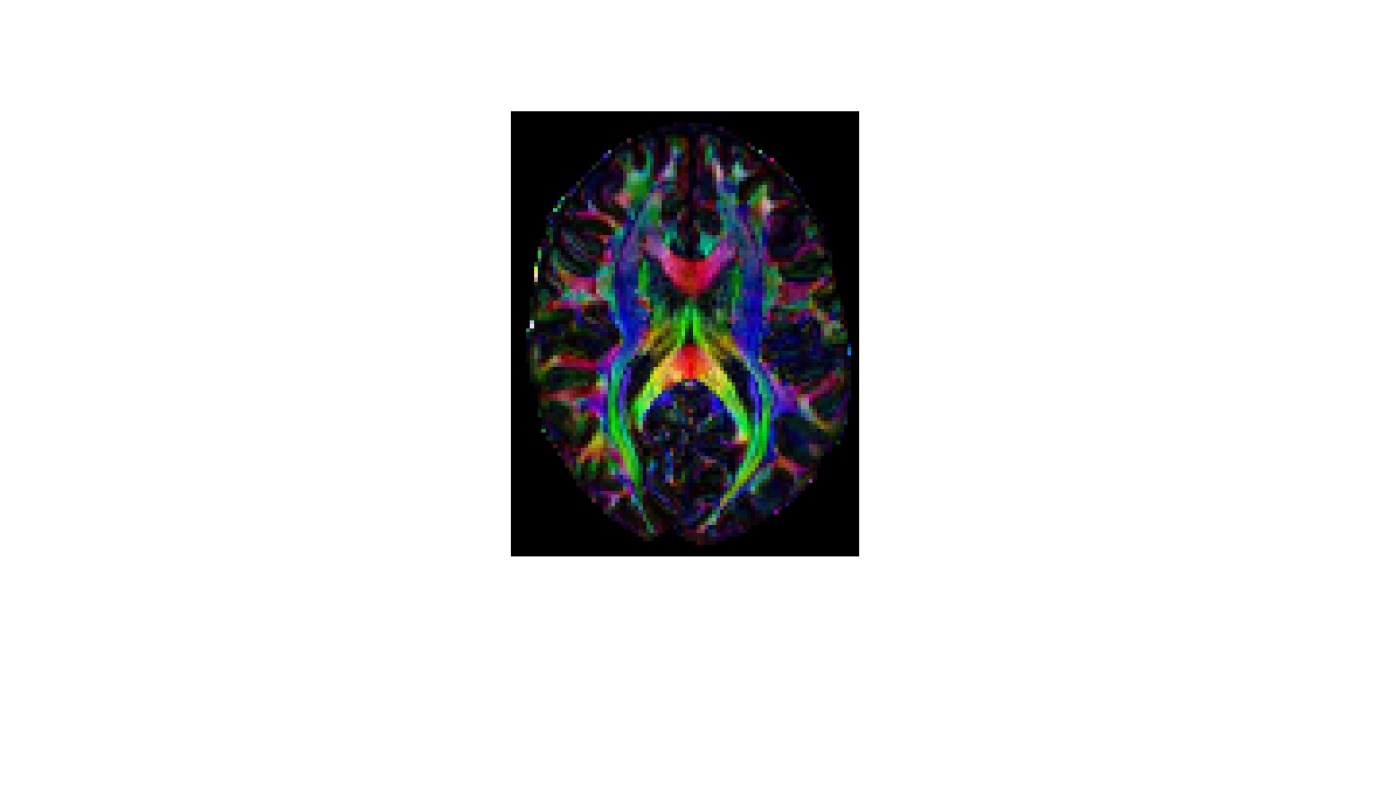


Figure 3 Principal diffusion direction map

To determine the principal diffusion direction (the direction of strongest diffusion), we observe the eigenvector  associated with the largest eigenvalue , then use Matlab to produce an image with which it can be visualised.

We use coordinates of (the Eigenvector associated with our largest Eigenvalue) to determine the red, green, and blue pixel intensities, and scale by FA to control brightness.

Thus each voxel is assigned a colour based on the principle direction of diffusion and brightness based of the directionality / relative magnitude of that diffusion. This increases the visibility of the information, see figure 3. =

## Dealing with Noise

To improve the accuracy of the estimation, we take into account noise surrounding the brain scan and remove unwanted data from our calculations.

Negative values in the dataset cause additional problems (particularly as we take the logarithm). To combat this, we can note that for S, the measure of signal intensity, “direction” is irrelevant. Thus, as we are interested in only magnitude, we use absolute values of and to remove negative readings.

To make the edges of the scan more interpretable, we apply a binary mask. This filters out irrelevant scan data by identifying actual brain tissue and excluding data outside this region from our calculations.

Despite these two techniques for reducing noise, the final Mean Diffusivity figure still displays some unwanted artifacts. This is partly due to the nature of the analysis - using an average value predisposes the image to a low contrast. Thus, a display threshold of 10% of the maximum value is put in place. This effectively increases the clarity and overall visualisation of the image.

|  |  |
| --- | --- |
| Low contrast Initial MD output | Higher Contrast output using threshold |
| A close-up of a brain  Description automatically generated | A close-up of a brain  Description automatically generated |

# Part II: Feature Extraction

The term ‘eigenface’ is commonly used within facial recognition and detection, (Acar, 2021), and can be described as a key facial feature, but to be used to its full potential there is typically a collection of eigenfaces, each representing key features of a face, for example one has bushy eyebrows, one has a thin nose, but most importantly in this case, one or more can represent having a moustache. Each eigenface is just an eigenvector ascribed to their own ‘face’, with their selection based on their difference from the average face.

To calculate or ‘find’ these eigenfaces, the provided face data is converted into the form of columns within a matrix. This matrix’s rows are then averaged, providing the average face within the given data set, the visualisation can be seen as figure 4.



Figure 5 A visual array of Eigenfaces



**Figure 4, The average or 'mean' face of the data**

Typically, a compact, or economical singular value decomposition (SVD) is then applied to the difference between the average face and face data matrix, giving us U, Sigma, and V and reducing the matrix, ordering each face from most to least visually different based on the values of sigma, but for this data set there are 36 different faces from different angles, so the process is slightly tweaked. Each ‘person’ a collection of 29 image, so these 29 images are averaged, and these averages are attributed to their own columns in matrix ‘B’. The compact SVD is applied to the difference between matrix B and the average face, creating our data sets eigen faces. A visual array is then created of the first 20 eigenfaces, see figure 5 above.

For the rudimentary moustache detector, the approach is to check if the face fuifill two requirements. The brightness of grayscale image ranges from 0 to 255, where 0 is black and 255 is white. The first requirement is how much darker the moustache area on average is compared to the moustache area of the mean face. The second requirement is how much darker the moustache area is on average compared to the entire face on average. The first requirement is to filter out over exposed faces, as they tend to be brighter in the moustache area as well. The second requirement is to filter out darker, or overly shadowed faces that do not necessarily have a moustache but are darker in the moustache area compared to the mean face.

For each face, the average brightness of the pre-determined moustache area, the average brightness of the face, the difference between the average brightness of the moustache area and face, and the difference between the average brightness of the moustache area and average face are all calculated. If the moustache and mean face have a difference greater than x and the moustache and particular face have a difference greater than y, then the face is considered to have a moustache and added to an array. This array is the collection of every image within the data set that has been classified to have a moustache, it must be noted that bad lighting and darker skin tones can lead to mis-categorised images.



Figure 6 Faces the program deemed to have moustaches

Figure 6 is the result of checking all 1000 faces and displaying all faces that fulfil the requirements. Most faces either have moustaches or have shadow in the moustaches area due to lighting conditions. It is notable that the collection have no noisy faces, due to the second requirement filtering them out, and no over exposed faces due to the first requirements.

The limitation to this method of checking for moustache is that it cannot detect white moustaches, and the detection is heavily affected by lighting conditions. These can be resolved by adding additional conditions and taking average of multiple of the same face to minimize effect of bad lighting.

# Conclusion

In the context of interpreting MRIs using technology, and in this case MATLAB, it is possible to utilise this technology to interpret and map MRIs in different ways so they can be used as tools to identify possible malformation within the brain, like lesions or cancers. In addition to this, the use of eigenvectors and eigenvalues within MATLAB code can be manipulated to identify key components of images across large samples, in this case moustaches on faces, but in the future could be used to identify malformations across multiple layers within an MRI brain scan.

# Reference list

Acar, N. (2021, August 28). Eigenfaces: Recovering Humans from Ghosts - Towards Data Science. Medium. <https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184>

Department of Health & Human Services. (2014, August). MRI scan. Better Health Channel. Retrieved June 1, 2024, from <https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/mri-scan>

Elster, AD 2009, DTI, Questions and Answers ​in MRI. Retrieved 15 May 2024, <https://mriquestions.com/dti-tensor-imaging.html>

Jiang, H, C.M, P, Kim, J, Pearlson, GD & Mori, S 2005, ‘DtiStudio: Resource program for diffusion tensor computation and fiber bundle tracking’, Computer Methods and Programs in Biomedicine 81 , vol. 81, no. 2, pp. 106–116. Retrieved 10 May 2023, <http://individual.utoronto.ca/ktaylor/DTIstudio_mori2006.pdf>

Moroney, T 2024, MXB201 Project Description, QUT Canvas. Retrieved May 2024, <https://canvas.qut.edu.au/courses/17944/assignments/164926>