# 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 <https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/mri-scan> . 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 it’s 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. Thus, 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/mm2) - 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. For the purposes of our model, we will take b as 1000 s/mm2 (a typical value); the only variable changing throughout the scan are the directions **.**

We aim to estimate **D**, 3 x 3 symmetric positive definite matrix, at each voxel, given a two-dimensional slice of a patient’s scan, from which we can extract and .

Substituting these values into the equation above, we can obtain a system of 64 equations (for 64 directions), which 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

and b is …

Since we cannot find *D* such that , we must find the most fitting solution for D, 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 can use MATLAB’s built-in Gram-Shmidt process (since we're using floating point arithmetic) which simply requires a backslash operator on the rectangular matrix A - outputting the diffusion tensor D:

We must take into account noise (corrupt or meaningless samples) surrounding the brain scan and remove unwanted data from our calculations to improve the accuracy. In particular, since we will be taking the logarithm, we require any negative values to be removed from the data set, as these are a product of taking measurements in practice using machinery. In order to input this, we take the absolute values of S and S0.

With regard to noise, we will use a binary mask in MATLAB to filter out the unwanted scan data, provided under the name “mask”. This identifies the actual brain tissue in the scan and removes data outside the scope of the brain.

In order to produce a more accurate figure of the Mean Diffusivity, we set a threshold of 10% of the maximum value. This reduces noise inside the brain in the map. (note: could include a before and after?)

From our symmetric matrix D, we can obtain eigenvalues and eigenvectors. In particular, a symmetric positive definite matrix has real eigenvalues and orthogonal eigenvectors.

Then the three common imaging techniques for diffusion tensor imaging are as follows:



*Figure 1 mean diffusivity*

## Mean diffusivity map

To determine the magnitude of the diffusion at each voxel, we find the mean diffusivity (mean of all three eigenvalues) and produce Figure 1.

## Fractional anisotropy

To determine the fractional anisotropy, (a measure of how the eigenvalues differ), we use the given formula (Elster 2009)

“For perfect isotropic diffusion . “Brighter areas are more anisotropic than darker areas” – Elster, 2009.

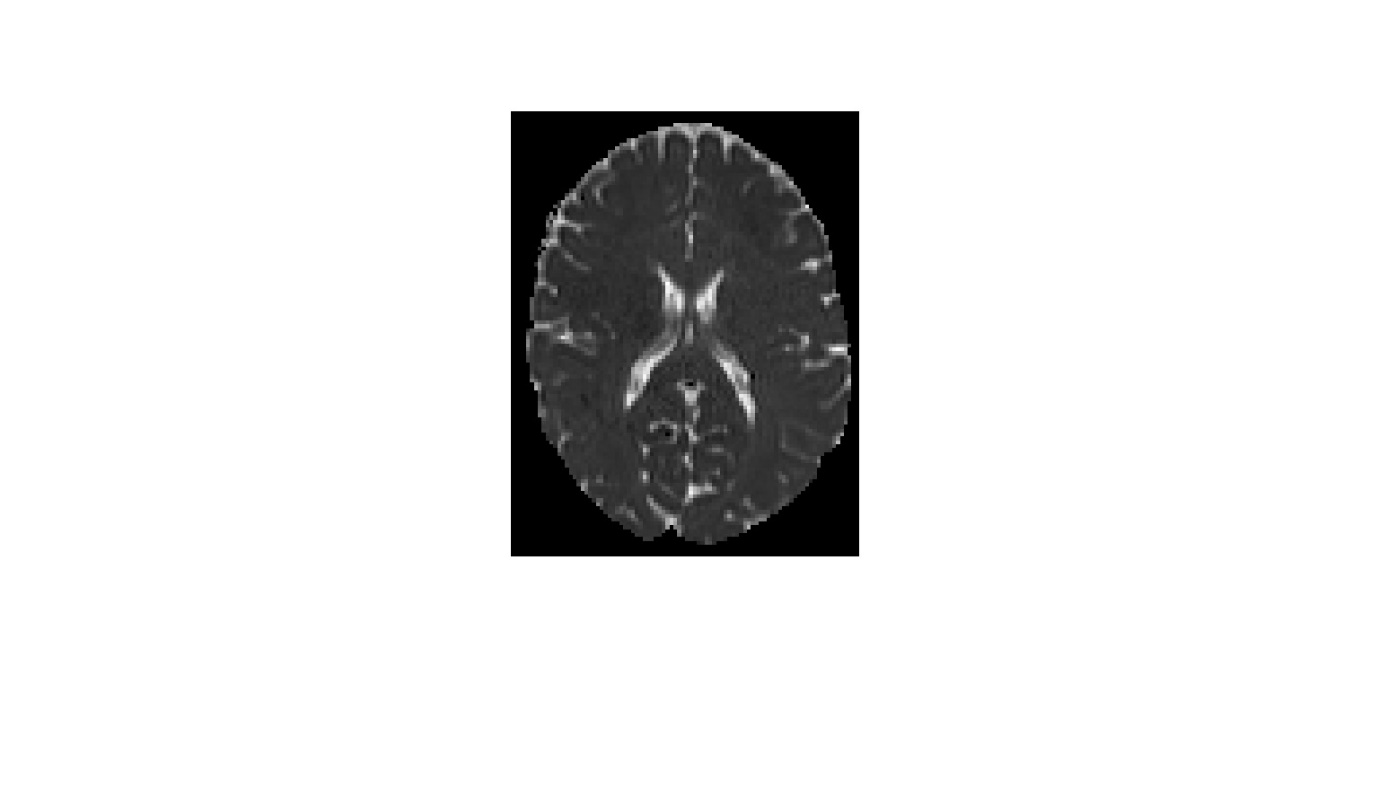


Figure 2 Fractional Anisotropy

Thus, from our image figure 2, we can see the very outside of the brain, and the x-shape in the inside, are much brighter and thus more anisotropic.

## Principal diffusion direction

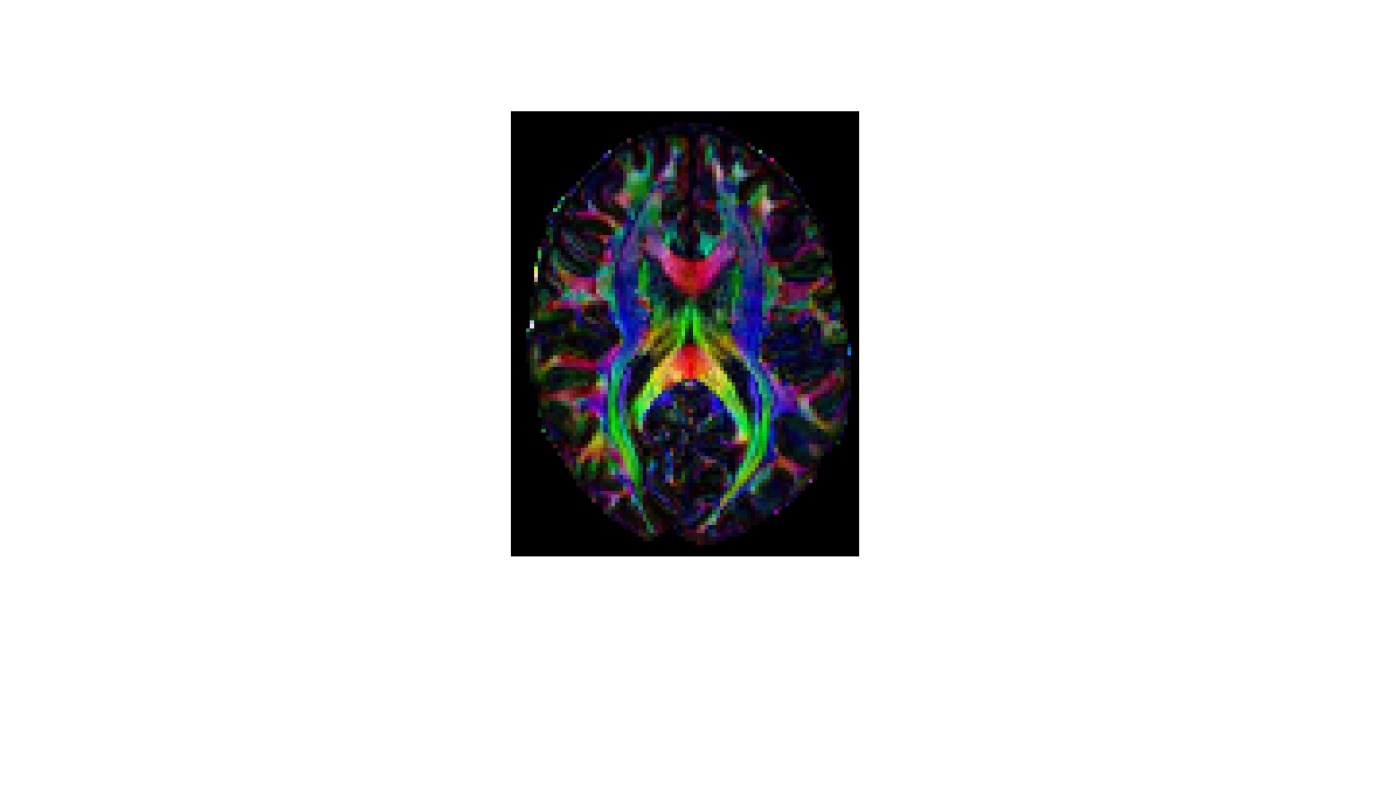


Figure 3 Principal diffusion direction map

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 both the anisotropy and direction

# Part II – Feature Extraction

The term ‘eigenface’ is commonly used within facial recognition and detection, <https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184> , and can be described as a key facial feature, but to be used to it’s 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 eigen faces, 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 with the given data set, the visualisation can be seen as Fig. 4.

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 and 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 VD 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 Fig 5.



Figure 4 A visual array of Eigenfaces



Figure 5, The average or 'mean' face of the data

To create a moustache detector, two values must first be defined. The first being how much darker the moustache area is compared to the mean face ‘x’, for this program a value of 5 is used, and a value for how much darker the moustache area is compared to the rest of the individual face ‘y’, for this a value of 8 is used, bear in mind the brightness is on a scale of 0 to 255. For each face the average brightness of the pre-determined moustache area, the average brightness of the particular 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 counted as having a moustache and added to an array, see complete array in Fig 6



Figure 6 Faces the program deemed to have moustaches

# Conclusion

# Reference list

Elster, AD 2009, DTI, Questions and Answers ​in MRI, viewed 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, viewed 10 May 2023, <http://individual.utoronto.ca/ktaylor/DTIstudio\_mori2006.pdf>.

Moroney, T 2024, *MXB201 Project Description*, *QUT Canvas*, viewed 1 May 2024, <https://canvas.qut.edu.au/courses/17944/assignments/164926>.