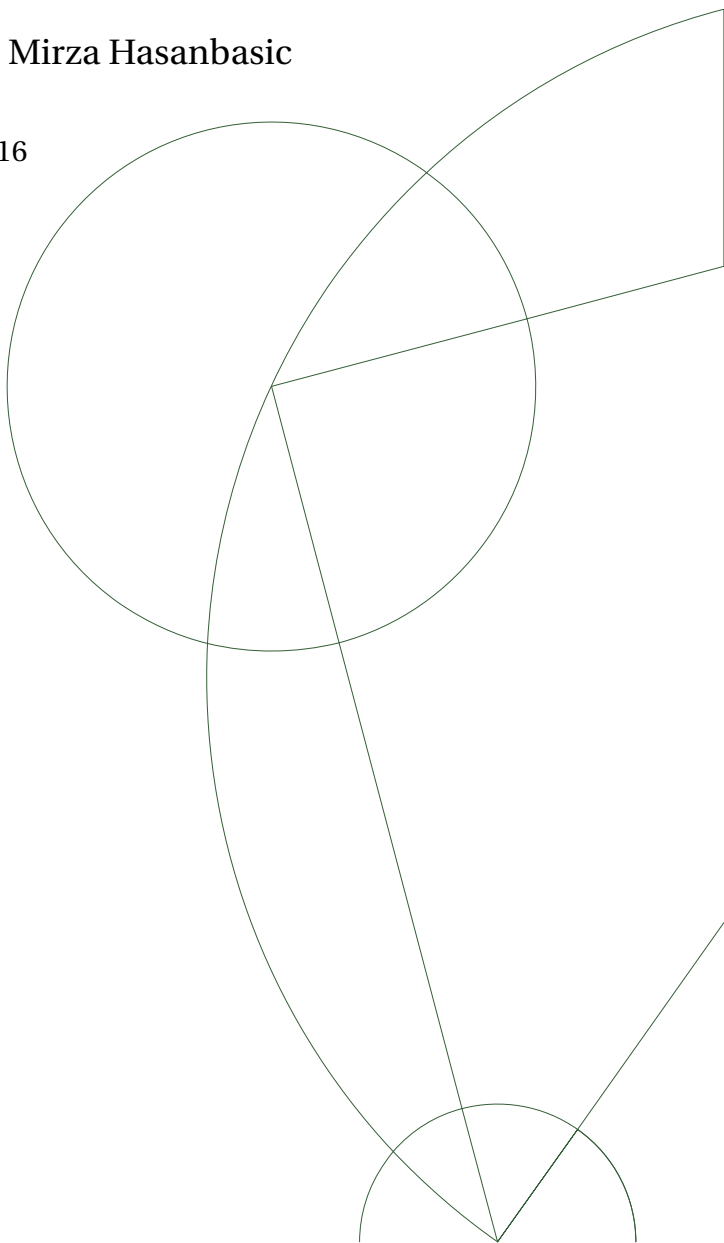
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

MR Image texture analysis applied to the diagnosis of Alzheimer's Disease

Mathias Bjørn Jørgensen & Mirza Hasanbasic

10th June 2016



Contents

List of Figures

List of Tables

List of Corrections

Mathias Bjørn Jørgensen & Mirza Hasanbasic

Abstract

This report will examine MRI scans of brains, using image texture analysis and machine learning

Chapter 1

Introduction

Alzherimer's Disease (AD) is the most common cause of dementia among people and is a growing problem in the aging populations. It has a big impact on health services and society as life expectancy increases. In 2010 the total global costs of dementia was estimated to be about 1% of the worldwide gross domestic product[1]. AD is the cause in about 60%-70% of all cases of dementia[1] and about 70% of the risk is believed to be genetic [2]. Currently there are no way to cure dementia or to alter the progressive course. But however, much can be done to support and improve the lives if AD is diagnosed in the early stage of progression [1]

In this report we will examine MRI data of the hippocampus using image texture analysis and apply machine learning in order to diagnose AD in patients. Our dataset contain 100 patients 50 control and 50 with AD.

We will be using two different image texture analysis method, one which will me in 2D[3][4] and the other one will be in 3D[5], from which we calculate the data to the gray level co-occurrence matrix (GLCM).

1.1 Problem Definition

Is it possible to classify MRI data of the hippocampus into groups of healthy controls vs Alzheimer's patient, using a predefined set of image texture metrices, with an accuracy greater than 80%?

Is there a difference in diagnosing AD successfully by calculating the co-occurence matrix in 3D compared to 2D.

[1] With the terms as of direct medical costs, direct social costs and costs of informal care

Chapter 2

Data

The MRI head scans were acquired on a General Electric 3-T for all our 100 patients, were we got 50 normal subjects and 50 AD patients.

Chapter 3

Method

3.1 Image texture analysis methods

Image texture is a feature that can help to segment images into regions of interest and to classify those regions. Textures gives us some information about the arrangement of the intensities in an image.

3.1.1 Co-occurrence matrix

The co-occurence matrix (COM) is second-order statistics methods, which is based on information about colours in pair of pixels. The matrix is defined over the image with distribution values at a given offset. Mathematically we have a COM matrix **C** which is defined over an $n \times m$ image **I**, with $\Delta x, \Delta y$ being the parameterized offset, is calculated by

$$C_{\Delta x,\Delta y}(i,j) = \Sigma_{p=1}^{n}\Sigma_{q=1}^{m} \begin{cases} 1, & \text{if } I(p,q) = i \text{ and } I(p+\Delta x, q+\Delta y) = j \\ 0, & \text{otherwise} \end{cases}$$

FiXme Note: find reference

FiXme Note: find reference

The element (5,4) in the COM can be translated to meaning how many times there exist an element in the image with GI **FiXme Note: level or intensity?** 5 and another element offset $\Delta x, \Delta y$ from the originial with greyscale intensity (GI) 4, i.e. if the offset is (0,1)**FiXme Note: rigtigt offset?** and the first element is (x,y)(4,3) with GI 5 it would mean that element (x,y)(5,3) would have GI 4. If COM(4,4) is ten, it translates into there being ten instances with element (x,y) = 5 and (x+Δx,y+Δy) = 4. COMs calculated on GIs are often called gray-level co-occurence matrices (GLCM).

FiXme Note: level or intensity?

FiXme Note: rigtigt offset?

A single image have multiple GLCMs as different offsets creates different relations. Consider a 3 × 3 matrix looking at element (2,2) we can then create eight different offsets, (1,0),(1,1), (0,1)),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1) **FiXme Note: ændre til** $COM_{(0,1)}$... **osv**, however they are not unique.

FiXme Note: ændre til $COM_{(0,1)}$... osv

Focusing on the two offsets (0,1), (0,-1) in element (2,2) and (1,2) with GI 1 and 2 respectfully increases the entry $\text{GLCMs}_{1,0}(1,2)$ and $\text{GLCMs}_{-1,0}$ (1,1) with one, showing that $\text{GLCMs}_{0,1}{}^{\mathbf{T}}$

= GLCMs$_{0,-1}$. There exist the same relation between (1,1)(-1,-1), (0,1)(0,-1), and (-1,1)(1,-1). This leaves four different offsets for analysis (0,1),(-1,1), (-1,0)),(-1,-1) in general (0,d),(-d,d),(-d,0),(-d,-d) where d is the distance which are commonly named angles 0°, 45°, 90° and 135°.

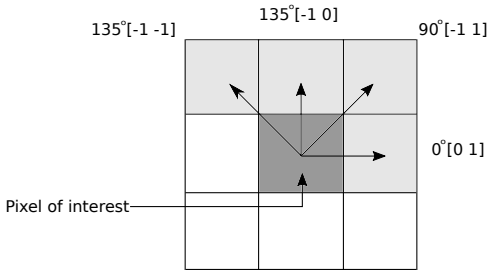


Figure 3.1: Example of the offsets for the 2D

The co-occurrence matrix is quadratic with the number of rows and columns equal to the amount of GI, for example if we have 256 GI we get a 256 × 256 GLCM.

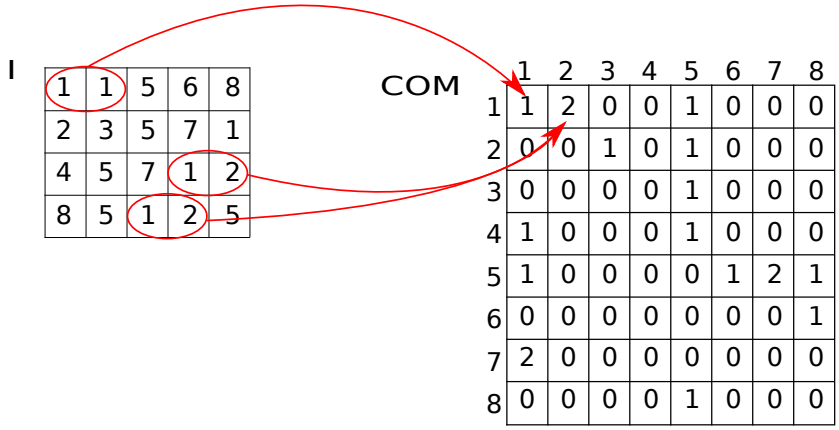


Figure 3.2: Example how the values in the GLCM are calclulated of the 4-by-5 image I. Element (1, 1) in the GLCM contains the value 1 because there is only one instance inf the image where two horizontally adjacent pixels have the values 1 and 1.

Extending this method to three-dimensions it is necessary to look on how the offsets are defined because the size of the GLCM is defined by the amount of GIs and not by the images it is derived from. Considering a 3 × 3 × 3 matrix we have a possible of 26 offsets. In two-dimensions it is possible to eliminate half of the offsets because of the relation GLCM$^{\mathbf{T}}_{d,d}$ = GLCM$_{-d,-d}$, and it is the same case in three-dimensions with the relation being GLCM$^{\mathbf{T}}_{d,d,d}$ = GLCM$_{-d,-d,-d}$. This leaves 13 offsets which are illustrated below.

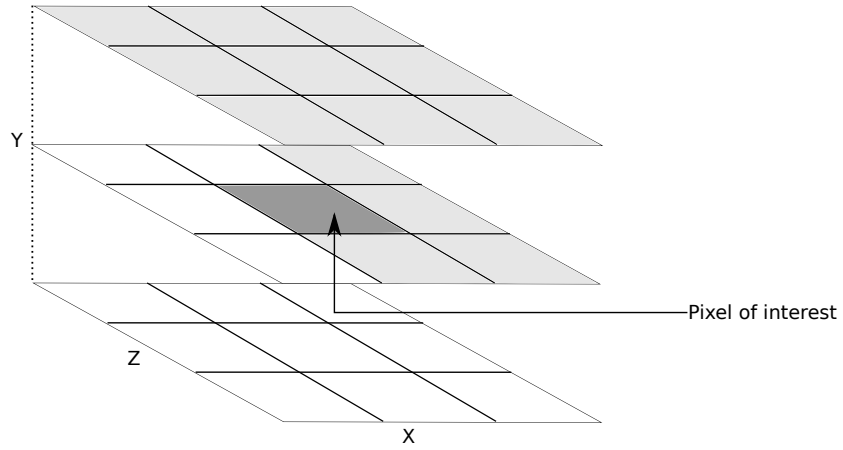


Figure 3.3: Example of the offsets for the 3D

To create a two-dimensional GLCM on three-dimensional image we create slices through the image. Given a $n \times m \times l$, n = 20, m = 20 l = 20, image we can create a slice for each n. Slice_1 is equal to the matrix M($n = 1 \times m \times l$), Slice_2 = M($n = 2 \times m \times l$), and so on, giving a total of 20 $m \times l$ images instead. It is possible for each slice to calcualte the $\text{GLCM}_{d,d}$ for each slice, and we define $\text{GLCM}_{d,d} = \Sigma_{n=1}^{2}0\ \text{GLCM}_{d,d}(\text{Slice}_n)$

These slices are done in all three directions of the image for the four difference offsets resulting in 12 GLCMs. There is an overlap between six of the GLCMs. Slicing through n axis with offset (0,d) is equal to slicing through the m axis with the same offset. Slicing through the m axis with offset (-d,d) is equal to slicing through the l axis with offset (-d,0). Slicing through the l axis with offset (-d,d) is equal to slicing through the n axis with offset (d,0), which is the transposed offset of (-d,0). Removing the duplicates leaves nine GLCMs which we calculate at distances d = (1,2,...,10), giving a total of 90 GLCMs for each mri scan.

The three-dimensional versions is also calculated for distances d = (1,2,...,10) resulting in 130 GLCMs for each mri scan.

3.1.2 Texture features from co-occurrence matrix

To compare the differences between the GLCMS 13 different features are computed. They are the same as used in freeborough from 1998 **FiXme Fatal: fix reference, lazy right now** except for one difference. The difference is how the correlation is calculated, correlation = $\sum_{i,j} \frac{(i-mui)(j-muj)glcm(i,j)}{(sigmai \cdot sigmaj)}$, where mu_i, mu_j, sigma_i, sigma_j are the means and standard deviations of Ci and Cj respectively. This correlation is called the Pearson product moment correlation coefficient and it determines how correlated a pixel is to its neighbour, over the whole image. cite We calculate two versions of these features, one where we normalized the COMs and one where we do not.

FiXme Fatal: fix reference, lazy right now

3.2 Machine learning

Short intro to Machine learning

3.2.1 10-fold cross-validation

Given a model with unknown paramers and a training set to which the model can be fit then the fitting process optimizes the models parameters to fit the training data. Validating this model against independent data (test data) from the same data pool as the training, it will generally turn out that the model does not fit the test data as well as the training data. It is known as overfitting and is a problem when the size of the training data set is small. Cross-validation is used to counteract overfitting.

Dividing the entire data set into 10 groups at random, one subsample is saved for testing and the reamining nine is used to fit the model. The procedures is done so all subsamples get to be used to validate exactly one time and the validation result can then be averaged to produce a single estimation. This solves the problem of overfitting, as the validation data set is never used in to fit the model. **FiXme Note: how does this remove overfitting?**

FiXme Note: how does this remove overfitting?

3.2.2 Feature selection

Feature selection is the process of selection af subset of relevant features for use in model contruction. The main goal of feature selection is to choose a subset of the entire set of input features so the subset can predict the output with an accuracy comparable to using the entire set to predict the output, but with a large reduction in the computationalcost. When a data-set contains many feautres that are either redundant og irrelevant it is possible to remove them without incurring much loss of information. Features may be redundant due to the pressence of another feature with which it is strongly correlated, while they may be very informative for the model their information have already been provided by a different feature. Only choosing a subset of the possible features have the added advantage ofr decreasing the complexity of the model.

3.2.2.1 Sequential Forward Feature

Sequential forward feature selection(SFS) is a greedy algorrithm to choose features. It starts off with finding the best possible single feature to describe the model. Given the feature set of that single feature, the next step is to find which other feature would improve the predictiveness of the model and then add that to the set of selected featuers. It continues to grow the set of selected features, until the goal have been reached. The goal can either be a specific amount of features, a specific accuracy for the model or it can stop if all choices of a new feature would decrease the accuracy. A problem with SFS is due to its greedy nature, there is no guarantee that the first feature selected is part of the optimal solution. Given three features X_1, X_2 and X_3 where X_1 is the best single feature, X_2 is second best and X_3 the worst, it does not necessitate that pairing X_1, X_2 is better than X_2, X_3, nor does it secure any other relation, meaning that the SFS does not guarantee the optimal solution.

3.2.2.2 Naive

3.2.3 K Nearest Neighbors algorithm

The K Nearest Neighbor (KNN) is a methoed that is used for classification and regression. It should be noted, that KNN is a non parametric lazy learning algorithm, which means that it does note make any assumptions on the underlying data distribution. In other words, it means that the training phase is fast and KNN keeps all the training data. It should be noted that KNN makes decision based on the entire training data set and in the KNN an object is classified by majority vote of its neighbors, with the object being assigned to the class most common amongst its *KNN*'s. 3.4
Since the training phase is minimal, then it should be noted that the testing phase is very costly for KNN in both memory and time and often it can be a worst case for time needed, since all points might take point in the decision making.

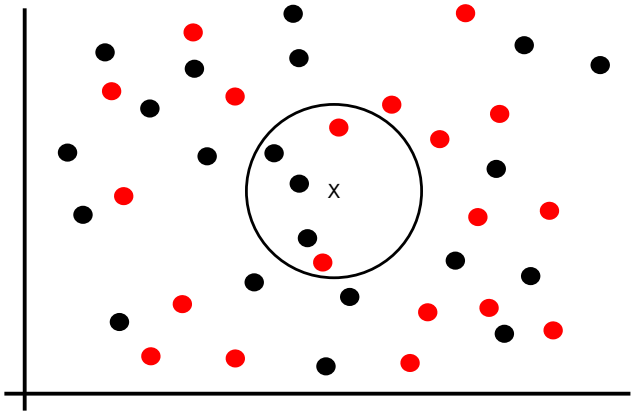


Figure 3.4: The KNN grows a spherical region until it encloses k training samples, and labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point $\mathbf{x}$ would be labelled the category of the black points

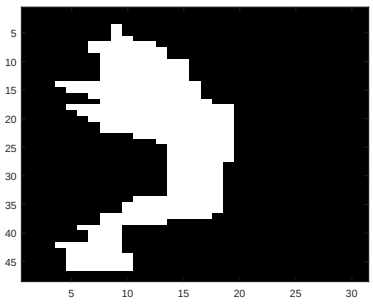
Because the KNN classifer predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters. Image that we have a large scale variabels, they will have a much larger effect on the distance between observations and hence the KNN classifier, than variables on a small scale. Often a good way to handle this problem is to standardize the data.

The disadvantage of KNN is that choosing a k may be tricky, so we are left to test the algorithm on multiple k's and often it needs a large number of samples for better accuracy.

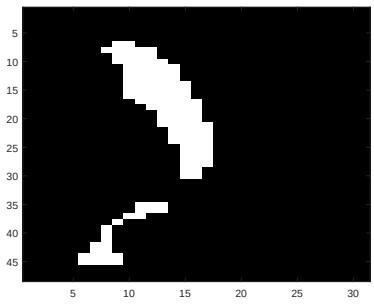
Typically will the KNN classifier be based on a distance, commonly it is based on the Euclidean distance between a test sample and the specified training samples. Let x_i be an input sample with p features $(x_{i1}, x_{i2}, ..., x_{ip})$, and n be the total number of input samples $i = 1, 2, ..., n$ and p the total number of features $j = 1, 2, ..., p$. The Euclidean distance between sample x_i and x_l, $l = 1, 2, ..., n$ is defined as $d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + ... + (x_{ip} - x_{lp})^2}$

3.3 Erode

Each patients hippocampus has been segmented in the MRI scan. The problems we can run into are that the background will blur with the segmentation i.e. the hippocampus. Performing a erosion can solve this problem and we can focus on the hippocampus, with the maximum number of details. As seen in figure 3.5a the erosion has not been performed yet, but we might have some problems with data surrounding the hippocampus is blurring out the edges of the hippocampus. To solve this, we create a mask to separate the hippocampus from the background data and end up with what is left in figure 3.5b



(a) Note eroded



(b) After erosion has been performed

Figure 3.5: Hippocampus at slice 10 on the X-axis as bitwise

The erosion we have used is called a city-block metric and can be seen in figure 3.6.

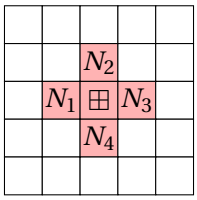
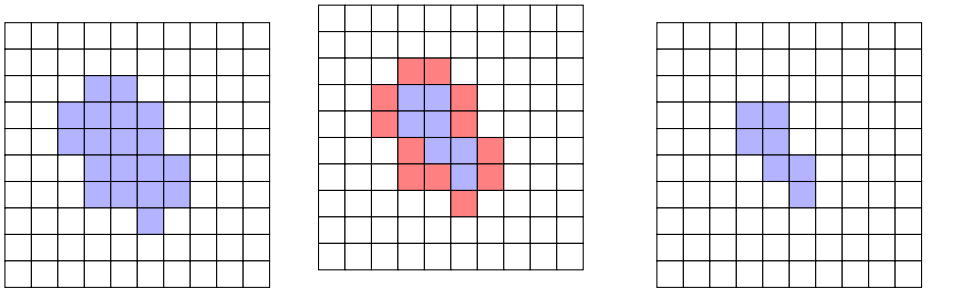


Figure 3.6: Text

To give an example of how the erosion works, it will be illustrated and we will use the city-block metric for this purpose. So we will use figure 3.6 and erode the image in figure 3.7.



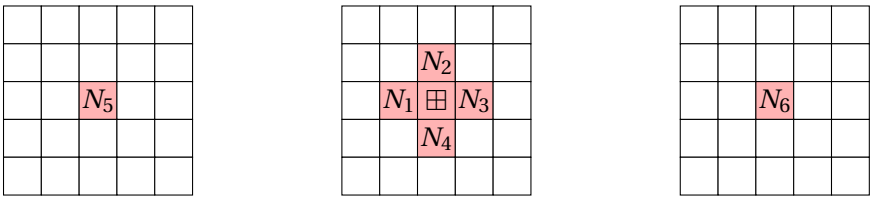
(a) Original image

(b) Performing erode on the image

(c) The image after erosion

Figure 3.7

As seen in figure 3.7 the noise (background) have been removed. This is an example in 2D. Now we wish to extend the erosion city-block to 3D. As seen in figure 3.6 it have 4 neighbours and when we extend this to 3D we will end up with 6 neighbours instead as seen in figure 3.8 and the concept is still the same as in 2D



(a) Text

(b) Text

(c) Text

Figure 3.8: Text

Chapter 4

Implementation

Chapter 5

Result

Chapter 6

Discussion

Chapter 7

Conclusion

Appendices

Appendix A

Co occurrence matrix derivation features

$$C_x(i) = \sum_{j=1}^{N} C(i,j)$$

$$C_y(i) = \sum_{i=1}^{N} C(i,j)$$

$$C_{x+y}(k) = \sum_{\substack{i=1 \\ i+j=k}}^{N} \sum_{j=1}^{N}, \quad \text{k = 2, 3, ... , 2N}$$

$$C_{x+y}(k) = \sum_{\substack{i=1 \\ |i-j|=k}}^{N} \sum_{j=1}^{N}, \quad \text{k=0,1,...,N-1}$$

$$f_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} \{C(i,j)\}^2 \tag{A.0.1}$$

$$f_2 = \sum_{n=0}^{N-1} n^2 \left\{C_{x+y}(k)\right\} \tag{A.0.2}$$

$$f_3 = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n} ijC(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{A.0.3}$$

$$f_4 = \sum_{i=1}^{N} \sum_{j=1}^{N} (i-\mu)^2 C(i,j) \tag{A.0.4}$$

$$f_5 = \sum_{i=1}^{N} \sum_{j=1}^{n} \frac{1}{1+(i-j)^2} C(i,j) \tag{A.0.5}$$

$$f_6 = \sum_{i=2}^{2N} iC_{x+y}(i) \tag{A.0.6}$$

$$f_7 = \sum_{i=2}^{2N} (i-f_6)^2 C_{x+y}(i) \tag{A.0.7}$$

$$f_8 = \sum_{i=2}^{2N} C_{x+y}(i) \log(C_{x+y}(i)) \tag{A.0.8}$$

$$f_9 = -\sum_{i=1}^{N} \sum_{j=1}^{N} C(i,j) \log(C(i,j)) \tag{A.0.9}$$

$$f_{10} = \text{variance of } C_{x-y} \tag{A.0.10}$$

$$f_{11} = -\sum_{i=0}^{N-1} C_{x-y}(i) \log(C_{x-y}(i)) \tag{A.0.11}$$

Bibliography

[1] Dementia. http://www.who.int/mediacentre/factsheets/fs362/en/. Accessed: 2016 April.

[2] Anne Corbett Carol Brayne Dag Aarsland Emma Jones Clive Ballard, Serge Gauthier. Alzheimer's disease. *Lancet*, page 13, March 2011.

[3] Peter A. Freeborough & Nick C. Fox. Mr image texture analysis to the diagnosis and tracking of alzheimer's disease. *IEEE*, 17(3):5, June 1998.

[4] L.M. Li F. Cendes G. Castellano, L. Bonilha. Texture analysis of medical images. *Neuroimage Laboratory*, page 9, April 2004.

[5] Sanjay Kalra Rouzbeh Maani, Yee Hong Yang. Voxel-based texture analysis of the brain. *Plos One*, page 19, March 2015.