University of Birmingham

School of Computer Science
Final year project

Zernike Polynomial for quantification of Macular Pigment

Author: Piers Allen (1150595)

MSci Mathematics and Computer Science

supervised by
Professor Ela Clarridge

Contents

Acknowledgements:

I would like to thank my family and friends for their patience and support throughout the project. I would also like to thank my supervisor Professor Ela Clarridge for her ongoing guidance and encouragement.

Abstract:

Purpose: To develop and evaluate a new method for characterising the distribution of macular pigment (MP) in the retina using Zernike polynomials for quantification of the pigment.

Methods: Four optimisation techniques were developed to solve the optimisation problem of finding the coefficients of the best fitting Zernike Polynomials given an image generated from the multispectral retinal image analysis(MRIA) tool. Using the coefficient generated from the optimal optimisation algorithm two classification techniques are explored to try and identify correlations between the distribution of MP and age/disease status.

Results: Classifiers producing low error rates when splitting the images by age or disease status were found using a data set of 106 MRIA images split into training and testing data sets. Both error rates were low enough to suggest a correlation exists in both situations, however, when the classifier attempted to split between both age and disease status at the same time the error was much larger.

Conclusion: The evaluation of the techniques used and conclusions reached may indicate that the techniques developed, in future, be used as a diagnostic tool in the future to help in patient diagnosis.

All software can be found at:
`https://git-teaching.cs.bham.ac.uk/mod-40cr-proj-2016/pxa395.git`

Keywords: Macular Pigment, Age-related Macular Degeneration, Macular Retinal Image Analysis, Zernike Polynomials, Classification, Optimisation.

1 Introduction

The objective of this project was to develop and evaluate a new method for characterising distributions of macular pigment. This retinal pigment is of interest because it has antioxidant and protective properties that may reduce the risk of progression of age-related macular degeneration (AMD).

Macular pigment (MP) is a yellow pigment that is found in the center of the retina. It has a peak at the centre of the fovea and declines with increasing eccentricity. Its main roles are protecting the macula from harmful blue light, and helping to maintain the function of the macula. AMD is a condition that can result in blurred vision in the centre of the visual field. "There is no cure or treatments for vision already lost" (Wikipedia, Macular Degeneration, 2017) so early diagnosis is significant in the treatment process. If correlations can be found between the distribution of MP and someones age / disease status this could be developed into a diagnostic tool. The few documented methods for measuring an individuals MP are explained below.

1.1 Current measurement tools and previous research

1.1.1 Heterochromatic flicker photometry (HFP) & Macular pigment optical density (MPOD)

HFP is a tool that uses "spectral absorption properties and retinal location of MP in order to measure macular pigment density (MPOD)." (Howells, Olivia, Frank Eperjesi, Hannah Bartlett, 2013). It works by "presenting a light stimulus of two alternating wavelengths at the fovea and at an eccentric retinal area". (Howells, Olivia, Frank Eperjesi, Hannah Bartlett, 2013) The result is a single number (MPOD).

1.1.2 Multispectral retinal image analysis (MRIA)

Researchers at the School of Computer Science, Birmingham, developed a new imaging method called multispectral retinal image analysis (MRIA) which generates 2-dimensional maps of macular pigment throughout the retina. [Styles, I.B., Calcagni, A., Claridge, E., Orihuela-Espina, F. and Gibson, J.M., 2006] This provided an opportunity to characterise the distribution of MP, not just the value at its peak.

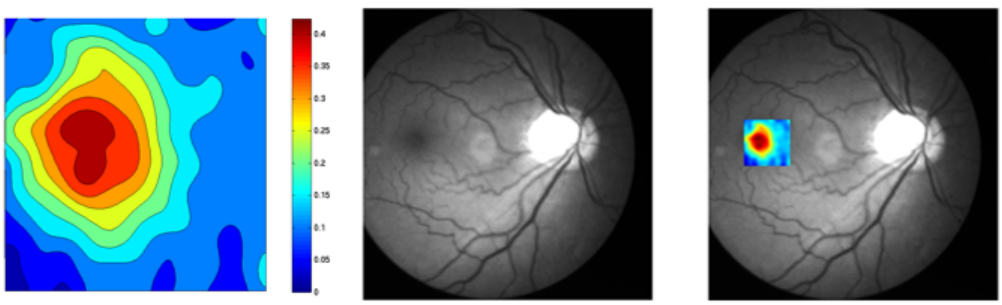

Figure 1: Example of the MP map. Left-to-right: MRIA MP map; Retinal image at 552nm; MP map overlaid on the retinal image.

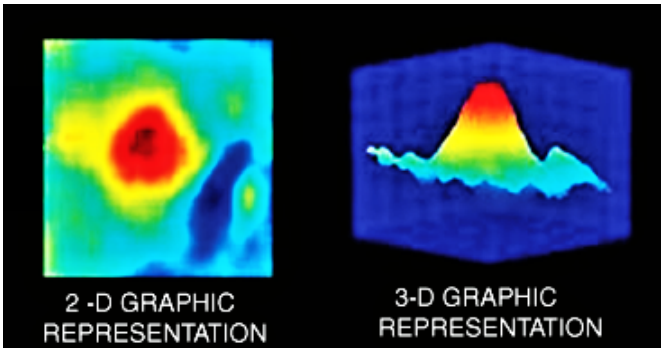

Figure 2: 3D representation of an example of a MP map

1.1.3 Fitting Gaussian distribution to the MP data:

A previous attempt to identify correlations between MP levels and age related macular degeneration used a method of fitting a Gaussian distribution to the MP data. The rest of this section is taken from an unpublished report to The Dunhill Medical Trust that explored this process.

"Computed MP maps were visually inspected to assess the levels and distribution of the pigment. Most distributions were observed to have a shape of two-dimensional Gaussian G which can be mathematically described as follows:

$$G(x, y; \sigma) = B + Ae^{-[\frac{(x-x_0)^2}{2\sigma^2} + \frac{(y-y_0)^2}{2\sigma^2}]} \tag{1}$$

where A is the amplitude, B is the base, σ is spread of Gaussian. Coordinates x_0, y_0 are the location of the peak which is not always locate at the centre of the fovea. After fitting a two-dimensional Gaussian function to the shape of MP distribution five numerical quantities were calculated from the maps (figure below): maximum MP level, absolute fitted peak height, fitted peak height in relation to the background, fitted peak full-width at half-maximum (FWHM), and fitted peak volume. These quantities were then used in statistical analyses to establish which ones provide the most useful clinical information."

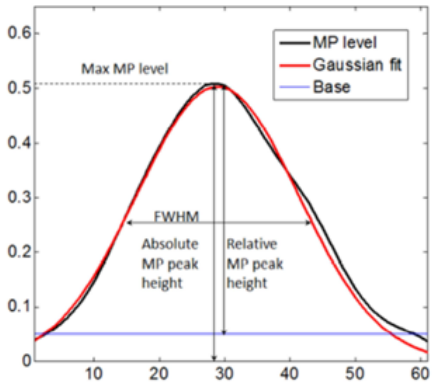

Figure 3: Example Gaussian Fit

"The objective of this study was to investigate whether multispectral retinal image analysis (MRIA), a new technique for mapping retinal pigments, is useful for measuring levels and distribution of MP to find differences between individuals with no clinical evidence of macular pathology and those diagnosed with Age-related Macular Degeneration (AMD). The study examined MP in three different groups; (1) those aged under 50 years without AMD, (2) those aged 50 years and over without AMD, and (3) those aged 50 years and over with AMD."

"The maps of MP were computed from 3 x 3 mm regions of interest (81 x 81 pixels in the image) centred at the fovea. Indices characterising MP distribution were computed both for individuals and for the three subject groups."

Individual measurements, whether for MPOD or for MP maps, showed no correlations with age or with AMD (see the plots from the Dunhill Medical Trust report).

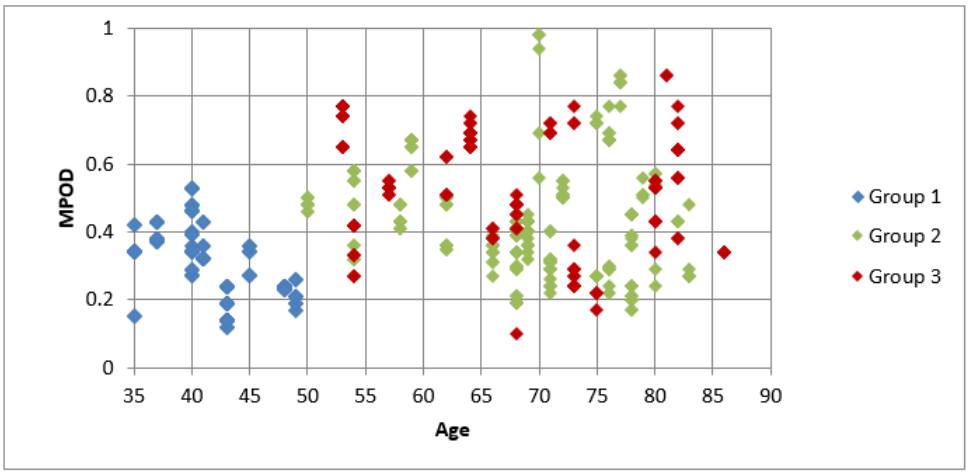

Figure 4: Pooled results on using the MPOD method

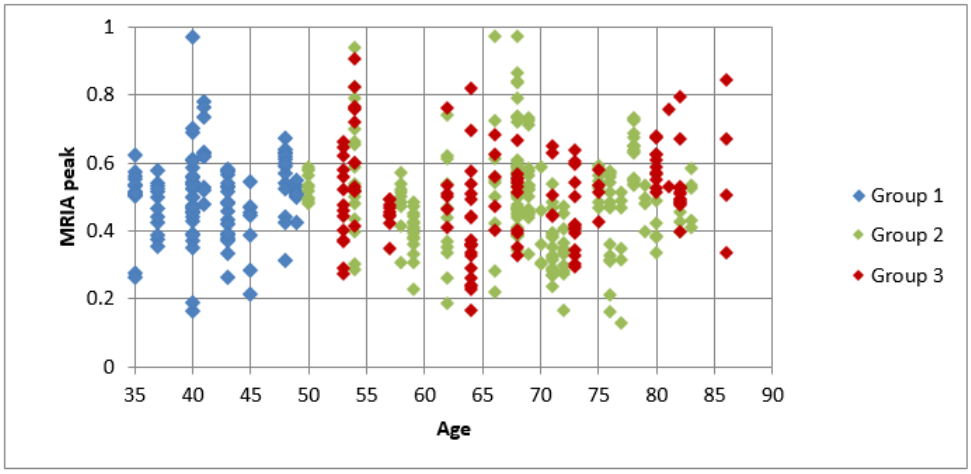

Figure 5: Pooled results on using the MRIA method

"Pooled results from the three groups suggest that the overall levels of MP are on average higher in healthy under-50 individuals than those over-50 with or without AMD. MP distribution might be more irregular in the over-50 groups than in the younger group. The correlation between age and MP levels was weak as measured individually by both techniques." (Dunhill Medical Trust report, unpublished).

1.2 Project specification

This project investigates the possibility of using Zernike Polynomials (ZP) as a method of characterising the distribution of MP from MRIA maps. Similarly to the 2D fitted Gaussian, the coefficients of ZP's can characterise the overall magnitude of MP distribution. Unlike 2D Gaussian parameters, additionally they have the ability to characterise fluctuations in the MP distribution. This project investigates multiple different optimisation algorithms to compute the quantification of the MP distribution. Using the quantification technique and a number of different classification algorithms, it will seek to identify correlations between the MP distribution and a patients age / disease status.

The scope of the project will be to do the following:

- Test multiple optimisation algorithms for the quantification of the images.
- Produce a program that uses an optimal optimisation algorithm to compute the quantification of a given image and returns a coefficient vector that can be used to recreate the image.
- Complete a number of experiment looking for correlations between sets of images using multiple classification techniques.
- Compare classification methods and produce a tool that can be used to classify future images with the optimal method.

1.3 Zernike Polynomials

Zernike polynomials [Zernike F, 1934] are being used as a basis function within this research. Previously they have been used in optics. However, here they are used to reproduce a 2D image of macular pigment distribution. "Any sufficiently smooth real-valued phase field over the unit disk can be represented in terms of its Zernike coefficients" (Wikipedia, Zernike Polynomials, 2016). These coefficients are used for classification of the images. There are two types of basis function, even and odd. A basis function is even when m is positive, and odd when m is negative. They are formulated slightly differently.

$$Even \;\; Functions : Z_n^m(p, \varphi) = R_n^m(p) cos(m, \varphi) \tag{2}$$

$$Odd \;\; Functions : Z_n^{-m}(p, \varphi) = R_n^m(p) sin(m, \varphi) \tag{3}$$

The variable p is the radial distance where $0 \leq p \leq 1$ and φ is the azimuthal angle. All basis functions have the attribute of being limited to a range of -1 to 1 on all axis (x,y

& z), however the actual data is limited to a circle of radius 1 centred at (0,0) along the x and y axes. The radial polynomial R_n^m are defined as follows.

$$R_n^m(p) = \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k (n-k)!}{k!(\frac{n+m}{2}-k)!(\frac{n-m}{2}-k)!} p^{n-2k} \tag{4}$$

Equations (2), (3), and (4) are all taken from [Eric C. Kinter, 1976].

Only certain combinations of integers form basis functions to Z so only correct pairings are used where m and n are both even or where m is odd and n is even. The equation produces a 2D array of data points that are strictly between -1 and 1. The first 15 basis functions are shown in figure 4, they have also been numbered for future reference.

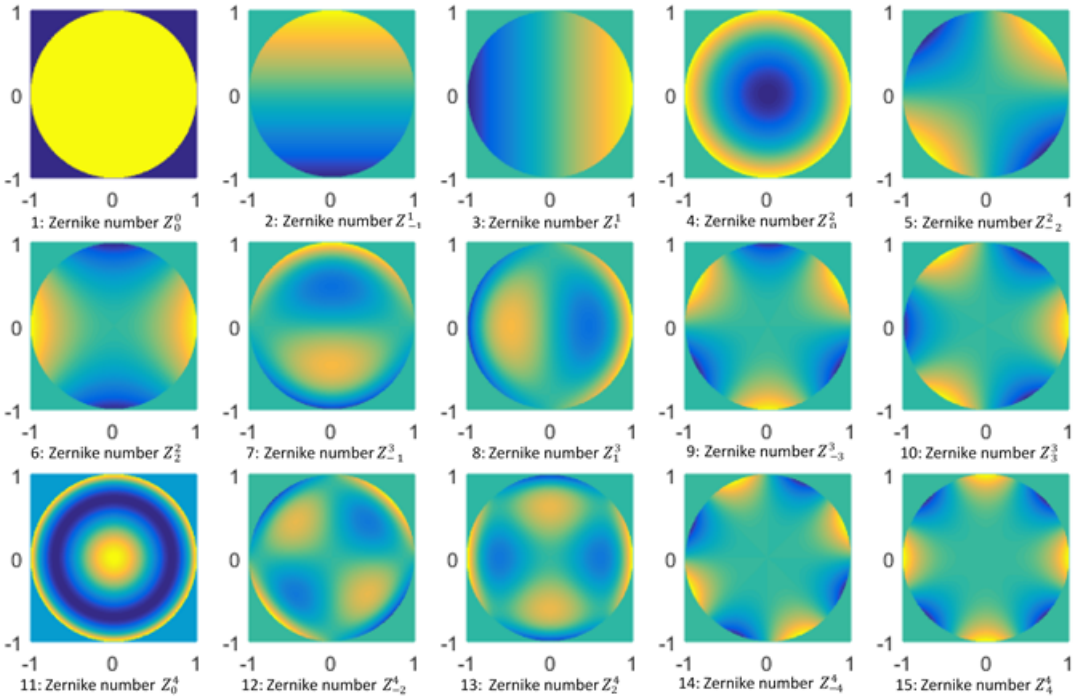

Figure 6: 2d representation of the first 15 basis functions

2 Using Zernike Polynomials to represents macular pigment data

2.1 Fitting Zernike Polynomials to Data

The first requirement of the project was to build a toll that could generate Zernike basis function. This was needed so that they could later be utilised by the quantification techniques. Matlab was used to design a function that would take in the two inputs m and n and produce a 2D map of the Zernike basis function. It became apparent that creating a library of the computed basis functions would be sensible. The function was adapted to have no inputs and produce a saved object containing the 2D maps of the first 105 basis functions, where the maps are the same dimensions as the images being used. Some examples of the 2D basis function maps are shown in figure 7.

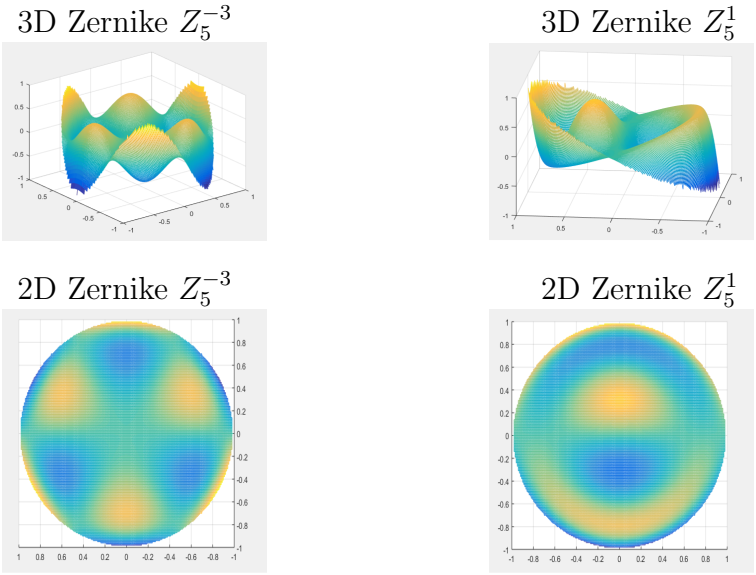

Figure 7: 3D & 2D representations of basis functions

To show how the library is used, an example of the combination of a coefficient vector and the basis function library is shown in figure 8.

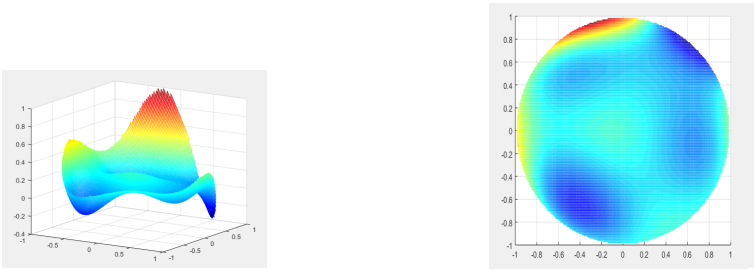

Figure 8: Example Combination of Coefficient vector and Basis function library

2.2 Pseudo Code for zernike calucation

Input: No input required
Ouput: Saved object contained the 2D maps of the first 105 polynomials.

```
#Set dimensions of the images. (how many data points)
dim = 256
#Create n & m arrays of n and m values in Z
n = [all n values]
m = [all m values]
#define the object that will hold all of the 2d arrays
Zernike = new array(105, 1)
for i = 1 to 105
    #init values
    x = -1, y = -1, counter = 1, counter2 = 1
    Z = new array(dim, dim)
    inc = 2 / (dim - 1)
    #calculate the zernike value for every point in the 2D data set
    while (x < 1)
        while (y < 1)
            #convert cartesian to polar for zernike calculation
            phi = arctan (y / x)
            p = y / sin (phi)
            #calculate the radial polynomial
            sum = 0
            for k = 0 to (n[i] - m[i])/2
                num = (-1)^k * (n[i]-k)!
                denom = k!* ((n[i] + m[i])/2-k)!*((n[i] - m[i])/2 - k)!
                sum = sum + ((num / denom) * p^(n[i]-2*k))
            end
            #check for negative & positive zernike & bounds
            if(m[i] < 0)
                mag = sum * sin(m[i] * phi)
            else
                mag = sum * cos(m[i] * phi)
            if(p > 1 or p < -1 or x == 1 or y == 1)
                mag = 0
            end
            Z(counter, counter2) = mag
            counter2++, y = y + inc
        end
        counter2 = 1, counter++, y = -1, x = x + inc
    end
    #Save the current 2D map
    Zernike = Z
end
Save Zernike
```

2.3 Optimisation Algorithms to Compute Quantification

The problem: given an image (MP map), find coefficients of the best fitting Zernike Polynomials. This problem is solved by optmisation that can be modelled as follows:

$$\min dist(I, f(y))$$

Where I is the original image, dist is the distance function being used, y is the coefficient vector and f() is the function that combines the vector with the basis polynomials.

The two distance functions that were used in this project are the Euclidean and Relative error calculations. The Euclidean distance is the sum of differences between two sets of point, in this case it will be the original image and the generated image. The equation is defined as follows:

$$\text{if } \mathbf{p} = (p_1, p_2, ..., p_n) \text{ and } \mathbf{q} = (q_1, q_2, ..., q_n) \text{ the distance is given by}$$

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

The other error that is explored is the relative error [Gene Golub, 1996]. The equation is defined below:.

$$\text{if } \mathbf{p} = (p_1, p_2, ..., p_n) \text{ and } \mathbf{q} = (q_1, q_2, ..., q_n) \text{ the distance is given by}$$

$$\forall p_i \in \mathbf{p} \text{ where } p_i \text{ doesn't equal 0, } d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{p_i - q_i}{p_i}$$

The following 4 algorithms were explored in this research to solve the optimisation problem:

- Quasi-Newton Algorithm (QN)
- Levenberg-Marquardt Algorithm (LM)
- Least Square fitting with Gaussian Elimination Algorithm (LS)
- Genetic algorithm. (GE)

Each of these algorithms was implemented in Matlab and then, when all were working, a series of tests was be performed to determine which was the most suitable. Both the LM and the QN were chosen as the optimisation toolbox within Matlab has a well documented package for both of these algorithms. Both algorithms also search for multiple maxima and minima, which is important in this optimisation as there are multiple peaks and divots within the image that the algorithms were trying to map to. The LS method was chosen because it could be solved incredibly quickly and provide a solution. However, it wasn't clear how accurate it would be. Finally, the GE method seemed a suitable option because if the mutations and combinations were of a good quality, it would gain a very accurate representation of the image. However, they can take a lengthy time to finish running. Explanations of all the algorithms & their implementations follow below. Testing to determine which was the most appropriate followed.

2.3.1 Quasi-Newton Algorithm

The Quasi-Newton method [Rob Haelterma, 2009] is designed to find both the maxima and minima of a given function. It finds the stationary points of the function which is where the gradient is equal to 0. The benefit of this method is that the hessian matrix of the function does not have to be computed. Instead, it is updated by performing analysis of successive gradient vectors. There have been a number of different methods for the analysis of the gradient but the method used in this project is the Broyden-Fletcher-Goldfarb-Shanno (BFCS) method [C Broyden 1970, R Fletcher 1970, D Goldfarb 1970, D Shano 1970]. Its iterative algorithm is defined below taken from [3] where f(x) is the objective function.

1. Obtain a direction $\mathbf{p}_k$ by solving $B_k\mathbf{p}_k = - \bigtriangledown f(\mathbf{x}_k)$.
2. Perform line search to an acceptable step size α_k in the direction of $\mathbf{p}_k$ then update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$.
3. Set $\mathbf{s}_k = \alpha_k\mathbf{p}_k$.
4. $\mathbf{y}_k = \bigtriangledown f(\mathbf{x}_{k+1}) - \bigtriangledown f(\mathbf{x}_k)$.
5. $\mathbf{B}_{k+1} = \mathbf{B}_k + \dfrac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} + \dfrac{\mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^T\mathbf{B}_k}{\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k}$.

The first step of the algorithm requires the inverse of $\mathbf{B}_k$ which can be obtained by using the Sherman-Morrison formula giving:

$$\mathbf{B}_{k+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{s}_k\mathbf{y}_k^T}{\mathbf{y}_k\mathbf{s}_k^T}\right)\mathbf{B}_k^{-1}\left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}\right) + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} \tag{5}$$

The above 5 steps and equation (4) are taken from [C Broyden 1970, R Fletcher 1970, D Goldfarb 1970, D Shano 1970, Wikipedia BFGS, 2016]

2.3.2 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [Kenneth Levenberg, 1944, Donald Marquardt, 1963](LMA) is an optimisation algorithm used to solve non-linear least squares problems. It is used mainly for fitting problems, and can also be used to find the local minimum, but not the global minimum. This method is often thought of as a combination of steepest descent and the Gauss-Newton methods. Given a set of m pairs of independent and dependent variables (x_i, y_i), it finds the parameter β of the curve $f(x, \beta)$ so the sum of the squares of the deviations is minimised.

$$\hat{\beta} = argmin_\beta \sum_{i=1}^{m}[y_i - f(x_i, \beta)]^2 \tag{6}$$

If it starts with an initial guess of the parameter vector $\beta = (a, b, ...n)$, after each iteration step a new estimate is made which is represented by $\beta + \delta$. δ is determined by the approximation of the linearisation of $f(x_i, \beta + \delta)$. The approximation is as follows:

$$f(x_i, \beta + \delta) \approx f(x_i, \beta) + J_i\delta \tag{7}$$

$$Where \;\; J_i = \frac{\partial f(x_i, \beta)}{\partial \beta} \tag{8}$$

The gradient will be 0 at the minimum of the points, so the first order approximation of $f(x_i, \beta + \delta)$ where $S(\beta)$ is the sum of squares gives:

$$S(\beta+\delta) \approx \sum_{i=1}^{m}(y_i - f(x_i,\beta) - J_i\delta)^2 \tag{9}$$

If the derivative of $S(\beta+\delta)$ is taken with respect to δ, setting the equation to equal 0 gives:

$$(\mathbf{J}^T\mathbf{J})\delta = \mathbf{J}^T[y - f(\beta)] \tag{10}$$

where $\mathbf{J}$ is the Jacobian matrix whose i^{th} row equals J_i and where f and y are vectors with i^{th} component $f(x_i,\beta) and y_i$ respectively.

Pseudo code from [Madsen et al, 2004]

Input: A vector function $f : R^m -> R^n$ with $n \geq m$ a measurement vector $x \in R^n$ and an initial parameters estimate $\mathbf{p}_0 \in R^m$.
Output: A vector $\mathbf{p}^* \in$ minimising $\|x - f(\mathbf{p}\|^2$.
Algorithm:
k:=0; v:= 2; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A}:= \mathbf{J}^T\mathbf{J}; \epsilon := \mathbf{x} - f(\mathbf{p}); \mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
stop:=$(\|\mathbf{g}\|_\infty \leq \epsilon_1); \mu := \tau \cdot max_{i=1,...,m}(A_{ii})$;
while(not stop) and (k $\leq k_{max}$)
    k:=k + 1
    repeat
        Solve$(\mathbf{A} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{g}$;
        if$(\|\delta_\mathbf{p}\| \leq \epsilon_2(\|\mathbf{p}\| + \epsilon_2))$
            stop:=true;
        else
            $\mathbf{p}_{new} := \mathbf{p} + \delta_\mathbf{p}$;
            $p := (\|\epsilon_\mathbf{p}\|^2 - \|\mathbf{x} - f(\mathbf{p}_{new}\|^2)/(\delta_\mathbf{p}^T(\mu\delta_\mathbf{p} + \mathbf{g}))$;
            if $p \geq 0$
                stop:= $(\|\epsilon\mathbf{p}\| - \|\mathbf{x} - f(\mathbf{p}_{new})\| \leq \epsilon_4\|\epsilon_\mathbf{p}\|)$;
                $\mathbf{p} = \mathbf{p}_{new}$
                $\mathbf{A}:=\mathbf{J}^T\mathbf{J}; \epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p}); \mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
                stop:=(stop) or $(\|\mathbf{g}\|_\infty \leq \epsilon_1)$;
                $\mu := \mu \cdot max(\frac{1}{3}, 1 - (2p-1)^3); v := 2$;
            else
                $\mu := \mu \cdot v; v := 2 * v$;
            endif
        endif
    until $(p > 0)$ or (stop)
    stop:=$(\|\epsilon_\mathbf{p}\| \leq \epsilon_3)$;
endwhile
$\mathbf{p}^+ := \mathbf{p}$;

2.3.3 Least Square Fitting using Gaussian Elimination Algorithm

This uses the concept of finding the minimal difference between a set of data points and a constructed mathematical model. If the modelling function is as defined below, where y is the measured outcome, f is the function, x is the experimental input and $\mathbf{p}$ is a vector of unknown variables $\mathbf{p} = \{p_1, ..., p_n\}$. The difference between the model and the expected measurement are called residual values and are defined as a function of $\mathbf{p}$.

$$r_i(\mathbf{p}) = y_i - f(\mathbf{p}, x_i) \tag{11}$$

And the total least square error is defined as:

$$e(\mathbf{p}) = \sum_{i=1}^{n} r_i^2 = \mathbf{r}^T\mathbf{r} \tag{12}$$

In computing a fitting problem the objective is to minimise the value of $\mathbf{p}$, resulting in the following optimisation equation.

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}}\, e(\mathbf{p}) \tag{13}$$

The simplest version of least square fitting is when the model is of a linear composition. However, in this project it is not linear and therefore a more complicated model must be considered. The model is represented by formula 10 using the same definitions as above and the residuals are defined in formula 11:

$$f(\mathbf{p}, x) = p_1 f_1(x) + ... + p_m f_m(x) = \sum_{j=1}^{m} p_j f_j(x) \tag{14}$$

$$r(\mathbf{p}, x_i) = y_i - \sum_{j=1}^{m} p_j f_j(x_i) \tag{15}$$

If $\mathbf{F}$ is defined to be the component $F_{ij} = f_i(x_i)$ where every row of $\mathbf{F}$ corresponds to one of the expected valued points and each column refers to a term within the model, and if $\mathbf{y} = (y_1, ..., y_n)$, the error can be re-written as:

$$e(\mathbf{p}) = (\mathbf{y} - \mathbf{F}\mathbf{p})^T(\mathbf{y} - \mathbf{F}\mathbf{p}) \tag{16}$$

As this is an optimisation problem where a model is being fitted to data, the minimisation of the error, e($\mathbf{p}$) is sought. Therefore it is necessary to find the value of the vector $\mathbf{p} = \mathbf{p}^*$ where $\mathbf{p}^*$ minimises e($\mathbf{p}$). To find the minimum, the error equation is differentiated with respect to $\mathbf{p}$ and the result is set to 0, this is then rearranged to get (16). This results in the equation below which can then be solved easily in Matlab using Gaussian Elimination \ operator (17).

$$\mathbf{p}^* = (\mathbf{F}^T\mathbf{F}\mathbf{p}^*)^{-1}\mathbf{F}^T\mathbf{y} \tag{17}$$

$$\mathbf{p}^* = (F' * F)\backslash(F' * y) \tag{18}$$

Equations labelled (10) to (17) were taken from [IB Styles, 2016].

Pseudo Code

```
#load the image & precomputed Zernike cells for all Zernike
#numbers up to the 105 degree.
Im = loaded image;
Z = loaded Zernike cells

#re size image to size of Zernike dimensions
(x,y) = dim(im)
(x2,y2) = dim(Z{1})
im = resize(im, x2/x, y2/y);

#reshape image to a vector of length x2*y2
I2 = reshape(im, x2*y2);
#reshape the Zernike cells to vectors and form a matrix
#of all of these vectors.
Z2 = new matrix(x2*y2, 105)
counter = 1;
for (Zernike z in Z)
    Z2(1,:) = reshape(z, x2*y2)
    counter++

#run linear Gaussian elimination
pstar = Z2'*Z2 \ Z2'*I2
```

2.3.4 Genetic Algorithm Solution

Genetic algorithms [Michael Vose, 1999] are an iterative process that takes an initial random population and progressively produces new populations based on the previous generation. They are known more specifically as meta-heuristic algorithms; this means that it is a problem-independent technique. Generally they are not a greedy method which allows them to explore more of the solution space and thus possibly gain a better solution. Every individual in a generation is referred to as a chromosome and is evaluated for its fitness. A genetic algorithm requires 2 things to run:

"1: a genetic representation of the solution domain" (Wikipedia Genetic Algorithm, 2016)

"2: a fitness function to evaluate the solution domain" (Wikipedia Genetic Algorithm, 2016).

The chromosomes used to breed the next generation are selected using the fitness function, so designing one that is efficient at calculating the optimal chromosomes is fundamentally important. For example, the fitness function designed for this optimisation problem calculates the error between the model produced using the chromosome's data and the image being quantified. The selection process for choosing the next generation in the iterative process can be achieved by various methods. Methods include Tournament Selection, Elitism and Stochastic Universal Sampling.

Once the selection has been made, a genetic manipulator is used to mutate the selected population into the next generation. The manipulator uses a number of different methods

to breed the next generation whilst maintaining genetic diversity so that the algorithm doesn't get stuck in local minima. The two ways of manipulating the population are: mutating a single chromosome; or breeding two chromosomes together. These are explained in a later section.

Once the mutations are complete, a new generation has been born which is then used to continue the iterative process. The iteration continues until either the maximum number of iterations it met or the fitness is below the required set value.

Selection Method Expanded

Tournament Selection

Tournament selection [Miller, Brad; Goldberg, David, 1995] involves running multiple tournaments among a selection of individuals chosen at random from the pool of chromosomes. The winner of each tournament is selected for breeding.

Elitism

This is the simplest but often the least accurate method of selection. The pool is sorted by its fitness and then the top N chromosomes are chosen to be used for the breeding of the next generation. The value of N is usually chosen based upon the population size. This causes the genetic algorithm to become slightly greedy in its selection which in turn means that rather than finding the global optima, it may find a local optima instead.

Stochastic Universal Sampling

Stochastic Universal Sampling [Baker, James E, 1987] works by choosing several chromosomes by repeated random sampling. It uses a single random variable to sample the entire population by choosing solutions at evenly spaced intervals. This means that the less fit members of the population have more of a chance to be chosen.

Genetic Manipulation Explained

Mutation

There are a number of methods that can be used for mutating [Marek Obitko, 2011] the selected population, including bit string mutation and flip bit. There are more methods that can be used, but they depend upon what the solution set is. For instance, if the set is a vector of values, multiplying random indexes of the vector by -1 would be a suitable mutation.

- Bit String mutation: The chosen solution is converted into its bit string representation and then altered. At random positions along the string the bit is flipped with the probability of 1/len, where len is the length of the binary string. This results in a rate of 1 alteration per mutation.

- Flip bit: The chosen solution is converted into its bit string representation and then instead of some of the bits changing, every bit is flipped to its opposite value e.g. 1 to 0 and 0 to 1.

Breeding

Breeding [Tomasz D. Gwiazda, 2006] is the idea of taking two or more selected chromosomes and combining them in a way that produces a new chromosome with some of each of the parent's characteristics.

- Single-point Crossover: This is where both parents' solution strings are split at the same point, once along them, and then the same half of each is swapped resulting in two new children.
- Uniform Crossover: this uses a fixed user defined ratio between two parents. For example, if the ratio is set at 0.5 then each child will have approximately half of the information of each parent. However, the cross over points are not defined and are chosen randomly. This still produces two children as an output.

2.3.5 Implementation

The genetic algorithm implemented in this project wass seeking to be optimise a vector of coefficients that are mapped to the Zernike basis functions which can be used to produce the quantified image. The limits of the algorithm were set as having max iterations of 1000(to limit the time taken by the process) and having a fitness value of less than $1 * E^{-4}$ (as this is within the bounds required for accuracy). The fitness function that was used is the same distance evaluation used for all of the algorithms.

The selection procedure used was elitism as this was the simplest to implement. However, if the algorithm was to be improved, this would also be the easiest place to start. The top 20 chromosomes in each working set were chosen to create the next generation of 200 values.

The genetic manipulator designed used six different kinds of combinations to produce the new generation. Three of them were breeding techniques which combined three different and randomly chosen parents of the top twenty chromosomes and then found the average of them in different ways. The other three were mutations: the first randomly incremented/decremented by a set value depending on the current best error; the second altered the sign of value at random indices of a randomly decided chromosome; and the third swapped values at different indices in the vector of a randomly chosen vector. Examples of all three follow.

- Mutation 1: $a = (1, 0, 1, 0, 0, 0, 1) \Rightarrow (1, 0, 1, 1, -1, 0, 1)$
- Mutation 2: $a = (1, 2, 1, 0, 1, 0, 2) \Rightarrow (-1, 2, 1, 0, -1, 0, -2)$
- Mutation 3: $a = (1, 2, 3, 4, 5, 6, 7) \Rightarrow (1, 2, 5, 3, 6, 4, 7)$

2.3.6 Pseudo Code

```
#generate your initial working data set by randomising
#n vectors.
workingSet = set(1000)
for (i = 1 to 1000)
    #vector of random values 105 digits long
    vec = random(105)
    workingSet(i) = vec

    #calculate the error for every vector in the working set
    vec(106) = error(vec)

#save the 20 best vector and proceed with them.
sort(workingSet)
top20 = workingSet(1:20)

numIteration = 1
#if the best result is within fitness end range or if max
#iterations has been reached.
while(numIteration != maxiteration or workingSet(1) > fitnessLimit)

    #run genetic manipulator on the top 20 vectors in working set.
    workingSet = geneticManipulator(workingSet);

    #sort new working set;
    sort(workingSet)
    numIteration++

end
return workingSet(1)
```

2.4 Algorithm Testing

2.4.1 Setup

To select the best algorithm to proceed with, a number of tests were performed on each one to determine which optimisation method produced the most accurate results. The same pre-generated images were used for every algorithm to ensure the test was fair. The methods (Levenberg Marquardt, LS, Quasi Newton and a Genetic Algorithm) were tested for accuracy, robustness and time taken to complete. Two different distance calculations, Euclidean and Relative were used. The Euclidean error suggested the total error over the whole data set and the relative error produced a percentage difference of the actual and generated image. Both inputs were used to determine accuracy.

Ten different vectors containing random coefficients were generated and proceeded to compute the data set using the image recreated tool detailed earlier. These data sets were used as images to feed into the optimisation algorithms. The results of the combined experiments produced sufficient data to finalise a conclusion as to which algorithm provided the best solution to the optimisation problem. The coefficients of the data used to create the images can be found in the appendix of the report, and can be used to recreate the experiment.

2.4.2 Tests on noise-free data

The first set of tests focused on the accuracy of the algorithms. Every image in the data set was fed into each optimisation algorithm and the error retrieved from the algorithm was saved. For each image & algorithm this was done twice to test both forms of error calculation.

Once all of the data had been collated the mean, median, minimum, and maximum error per pixel values were calculated for each algorithm and the tables in figure 9 were produced. The first set of results follow below and were used to see which of the algorithms proved the most accurate and took the shortest amount of time to compute. As accuracy was being tested there was no noise added to the images within this data set.

	Levenberg-Marquardt		Quasi-Newton	
	Euclidean	Relative	Euclidean	Relative
Mean	0.629236758	19.96236856	0.000745745	0.106974295
Median	0.021973953	5.680284543	0.0000109	0.110343903
Min	0.001373211	0.884293062	4.34E-09	0.05430632
Max	5.144813879	115	0.003671632	0.164500336

	Least Square with Gaussian Elimination		Genetic Algorithm	
	Euclidean	Relative	Euclidean	Relative
Mean	1.36778E-25	9.44731E-13	0.048983075	0.497121244
Median	4.12501E-26	6.3382E-13	0.048910778	0.497872287
Min	1.20552E-26	3.3709E-13	0.047795939	0.492376821
Max	9.41711E-25	2.60688E-12	0.049885285	0.499867195

Figure 9: Results of noise-free algorithm tests

To better visualise this table data, box and whisker graphs were produced. The box in the graph spans the interquartile range and the whiskers represent the maximum and minimum values of data range. These were useful to visualise the range and other characteristics of a large data set and to minimise the impact of any outliers in the evaluation of the algorithms. The graphs for no noise added data are in figure 11.

2.4.3 Tests on data with noise added

The second set of tests were carried out on images that had noise added to them. This process was done by using the 'noise' function in Matlab which takes in a array or values, in our case the 2D array of magnitudes, and randomly adds noise data to the whole set. In this case 10% noise was added to the data of each image. This was to test the robustness of each algorithm. This test cannot be repeated exactly due to the random nature of the noise function, but similar results would be expected if it was repeated. As before the same values were calculated and presented in exactly the same way. Tables and the graphs for noise added data are below and in figure 12 respectively.

	Levenberg-Marquardt		Quasi-Newton	
	Euclidean	Relative	Euclidean	Relative
Mean	18.8616371	157.9191157	4.036029203	12.41659751
Median	16.38106131	170.2040938	3.798643018	12.36533029
Min	3.616053957	13.21767553	2.548628522	11.6407333
Max	40.97913622	204.6990154	6.20405352	13.21767553

	Least Square with Gaussian Elimination		Genetic Algorithm	
	Euclidean	Relative	Euclidean	Relative
Mean	6.365587244	13.71269974	2.743673395	2.793788729
Median	6.032169127	13.40289971	2.673329435	2.746042629
Min	1.225308285	12.95882212	1.52314914	1.465946077
Max	12.7086698	15.83531849	3.766122808	3.747338206

Figure 10: Results of noise added algorithm tests

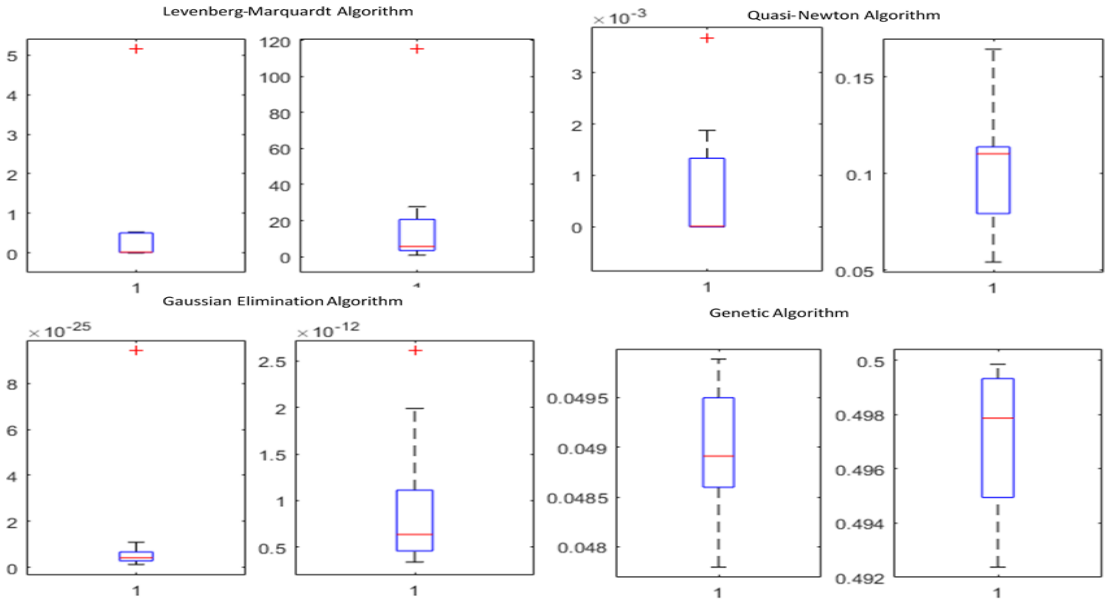

Figure 11: No added noise results off algorithm testing

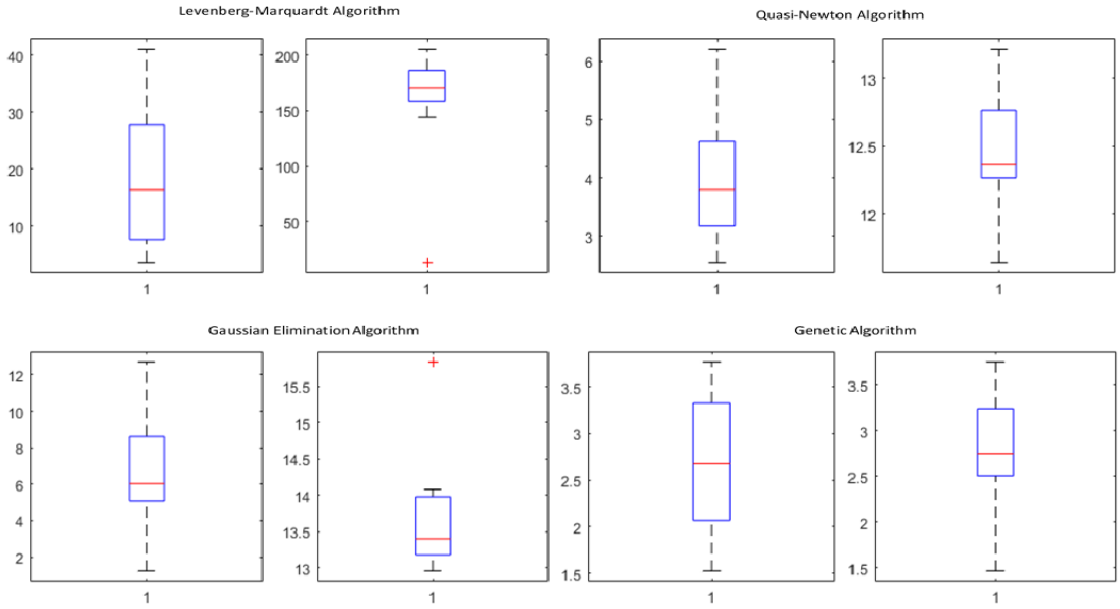

Figure 12: Noise added results off algorithm testing

2.5 Conclusion

Looking at the no noise data test, the most successful algorithm for the Euclidean error was the Least Squares with Gaussian Elimination algorithm. The range of the Euclidean error, even including the outlier, was as small as $1 \cdot 10^{-25}$, almost negligible when the values that are evaluated range between -1 and 1. This value compared to the other 3 algorithms, was vastly superior. The other 3 methods produced good results, the worst result, excluding outliers, being only a value around 0.5. As the results were so close the Euclidean error provides no additional information.

Comparing the relative errors, again the Least Square with Gaussian Elimination was the superior algorithm with a value in the range of $1 \cdot 10^{-13}$. However, looking at the other three algorithms, the Levenberg-Marquardt algorithm produced a bad error of around 10% to 20% compared to the $< 1\%$ of the other two. The preliminary conclusion that was made was that the Least Square with Gaussian Elimination was the optimal algorithm, and that the Levenberg-Marquardt solution had a weakness in accuracy. These conclusions were then compared with those from the data gathered from noisy images experiments. Initially the grading of the algorithms is as follows: 1. Least Square with Gaussian Elimination, 2. Quasi Newton & Genetic Algorithm, 3. Levenberg-Marquardt.

Looking at the robustness of the algorithms and firstly at the Euclidean results, all the algorithms except Levenberg-Marquardt produced similar results of a range of about 1 to 10. It was hard to distinguish to much from the Euclidean results as none stood out as optimal. Moving on to the Relative error, the best was clearly the Genetic Algorithm with an error rate of median 2.75% compared to the next best of 12 to 13% (Gaussian elimination and the Quasi-Newton approaches). The Levenberg-Marquardt result was well over 100% error, clearly not suitable.

	Noise-Free Data	Noise Added Data	Total Grade
Levenberg-Marquardt	4	4	8
Quasi-Newton	2	2	4
Gaussian Elimination	1	2	3
Genetic Algorithm	2	1	3

Figure 13: Table showing grades of algorithms over the different tests

With the choice reduced down to either the GE or the LS method, the running time of each method was now considered. The GE took approximately 40minutes per optimisation, whereas the Gaussian Elimination took 1-2 seconds. The Least Square with Gaussian Elimination was therefore superior and was chosen for this project. However, if the mutation and crossover methods within the GE were improved, it could prove a preferable solution. It might therefore be worth investigating the use of the GE again in the future.

3 Preliminary Evaluation of Classification Algorithms

The main objective of this project was to determine whether coefficient of Zernike Polynomials used to characterise the distribution of Macular pigment were able to classify individual images depending on age and the presence/absence of the disease (AMD). As explained previously, the subjects were divided into 3 classes: 1) Under 50 and no disease, 2) Over 50 and no known disease and 3) Over 50 and diseased. Two methods of classifying the data were tested: Support Vector Machines (SVM); and a K-Nearest-Neighbours (KNN) classifier. Once the images were quantified using the optimisation algorithm, a vector of the first 105 coefficients was produced to provide the input to the classification algorithms. The hypothesis was that only a small number of these coefficients are actually required to perform a successful classification. A number of methods were trialled to find out which combination of coefficients produced the best classifier. During the algorithms development phase a small subset of data (10 images) with known classifications was used.

3.1 Powerset Computation with K-Nearest-Neighbours Classifier(KNN)

Using the power-set of coefficients, the classifier was tested using all of the different combinations. The combination that produced the lowest training error was then chosen and used as the model for the classifier. Only this combination of coefficients was used for the classification of new images. The input of the user into the classifier was the value of k used and the number of coefficients in the quantification vector the algorithm has access to.

K-Nearest-Neighbours Classifier:

The following shows the break down of the algorithm into the training stage and the classification stage:

- Training: This consists of storing the set of data points and their classifications allocated for training the classifier.
- Classification: The classification of a point is decided using a user defined variable k, finding the k nearest neighbours employing a distance function, and then using either a weighted method or modal method to give the inputted point a classification.

Training:

To work out the optimal combination of coefficients for the given data set, a KNN classifier was trained using every combination of coefficients, and the training error was saved to see how accurate that choice of coefficients was. The training error was calculated by working out the number of training points miss-classified using the current classifier. Once all were calculated, the choice with the least training error was used as the training set. The number representing the coefficients used, and the defined k value was saved, along with only those coefficients from each item in the training set required (in order to save memory). This allowed the classifier to recognise which coefficients to draw from inputted data. The pseudo code for this is on the next page.

Classifying:

Firstly, the classifier checked that there was a saved KNN Classifier in the working directory, then loaded the data if there was. The image was then quantified over 105 coefficients using the optimisation algorithm. The classifier held a vector of coefficients required, which was then used to grab the correct coefficients from the quantification. The Euclidean distance function was used to find the defined k nearest neighbours and the mode was taken of the set of classifications of the neighbours. The modal value was returned as the classification of the image.

Pseudo Code:
Training the Classifier:
Input: Im as set of images set aside for training the classifier, class as their classifications, numCoeff as the number of coefficients being used and the set value of k
Ouput: Set of data and classifications

```
#find the quantified values of the images
coeffs = []
numData = length(im)
for i = 1 to numData
    #get the quantification
    coeff = LinearSquareOptimise(im(i))
    #save the firnst numCoeff digits
    coeffs(i) = coeff(1 to numCoeff)
end
#make the powerset of 1 to numCoeff and find its length
powerset = PowerSet(1 to numCoeff)
lenthPow = length(powerset)
minVal = (optimalCoeff, optimalError)
for i = 1 to lengthPow
    #use the powerset to define the required coeffs and
    #grab them from the data set
    reqCoeff = powerset(i)
    reqCoeffs = []
    for j = 1 to numData
        x = coeffs(j)
        reqCoeffs(j) = x(positions reqCoeff)
    end
    #for every data point calculate the classification
    #using the data set
    for l = 1 to numData
        #calculate all the distances
        distances = []
        for j = 1 to numData
            distances(j) = Euclidean(reqCoeff(l),
                                     reqCoeff(j))
        end
        #gather the k closest neighbours
        minDists = min (distances, k)
        classification = mode(minDists)
        if(classification != class(l)
            numMisClass + 1
    error = numMissClass / numData
    if (error < optimalError)
    minVal = (powerset(i), error)
end
return minVal
```

3.2 Powerset Computation with Support Vector Machines Classifier (SVM)

In addition to only trying the first 15 coefficients of the coefficient vector, an experiment was run using all the coefficients to establish whether the SVM could separate the classes using all the information available. Much like in the KNN, the powerset method was used to try every combination of coefficients during the training of the classifier. The user input in this case was only the number of coefficients the classifier had access to.

SVM Classifier:

The SVM classifier [Cristianini, Nello; Shawe-Taylor, John, 2000] is a supervised learning method that can be used for analysing data for classification and regression analysis. In essence the method constructs a hyperplane or set of hyperplanes in high-dimensional space that can be used to separate the data. As the data for this project was high dimensional, this section focuses on the non linear method for separation utilising a kernel function $K(x, y)$ selected to suit the problem. This approach transformed data into a different dimensional feature space where the hyperplane could be computed. The hyperplane is defined as: $\vec{w} \cdot \vec{x} - b = 0$ where x is the set data points satisfying this function, w is the normal vector to the hyperplane, and $\frac{b}{||\vec{w}||}$ determines the offset of the hyperplane along the normal vector $\vec{w}$

- Training: A training data set was inputted and used to compute the non-linear separating hyperplane, utilising the kernel approach.
- Classification: Using the function $\vec{w} \cdot \vec{x} - b$ where x is the input, the results were -1 or 1 which decided the class it should resided in.

Training:

The library within the Statistics and Machine Learning Toolbox was utilised to pass in a data set with their classes producing a trained classifier. For optimal accuracy, the same method as the KNN classifier was used with regards to the powerset of coefficients. This allowed the SVM to choose the optimal combination to classify the images. Once trained, the classifier was saved so it could be accessed at a later time.

Classification:

A new data point was passed in using a method from the same library above, which returned its classification using the saved classifier object. The testing error was calculated by the number of incorrectly classified points within the whole of the testing data set.

3.3 Experiments

To establish whether there was any baseline for a correlation between different sets of data, a small set of data with known classifications was used to run a test. The three sets for classification were: under 50 and healthy; over 50 and healthy; and over 50 and diseased.

3.3.1 Data

Ten images were used in this experiment. The images had known classifications and were a representative selection of all of the classes. The Zernike coefficient quantification of each of these images was computed and used as the inputs for the algorithms.

3.3.2 Classification Algorithm

Both the KNN and SVM methods were tested to establish whether both were are appropriate to use. The algorithms were not compared at this point. The KNN experiments were run first, and the SVM experiments followed.

3.3.3 Experiment - KNN

All possible combinations of coefficients were tested to see which produced the minimum error as defined in the algorithm. The value of k and the number of coefficients was changed to test which combination produced the lowest training error. Initially, the power set of all coefficients was calculated, then the algorithm was run on every combination calculating the error each time and storing it. The result of the experiment was the minimum training error computed and the coefficients that were used to classify the data.

3.3.4 Results - KNN

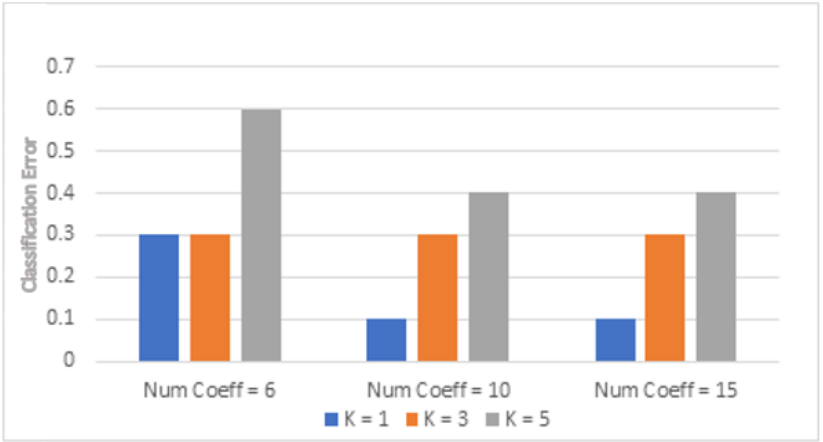

Figure 14: Graph represetning KNN Preliminary Results

3.3.5 Conclusion - KNN

There were several preliminary conclusions made from this experiment. The results showed that using a value of k equals 5 for all the different numbers of coefficients results in the highest error rate of the values of k tested. Therefore, k equals 5 was ruled out as a suitable value. For the remaining two values, the error results produced by k equals 3 were worse or the same as the error produced by k equals 1, so using k equals 1 appeared to be the optimal option. Assuming k equals 1, the lowest error was when the number of coefficients was 10 or 15. Using more coefficients merely increased the time complexity of the classification. As it provided no extra benefits, it seemed to be sensible to use only 10 coefficients.

The assumptions above led to a conclusion that the best combination of values was with k equals 1, and the number of coefficients equals 10 (with only a 10% error rate). As this experiment was only completed on a small data set the results the needed to be tested on a much larger data set to ensure the classification was still accurate. However, the preliminary results were promising.

3.3.6 Experiment - SVM

As before, all combinations of coefficients were tested. However, the number of coefficients was the only variable that could be altered in this algorithm. The method for checking all the coefficients and the way the results were gathered was the same as the previous experiment.

3.3.7 Results - SVM

Table representing results

Number of Coefficients	Classification Error	Optimal Coefficient Set
6	0.1	2,5,6
10	0.1	7,9,10
15	0.1	10,11,12,13

3.3.8 Conclusion - SVM

The only conclusion made from these results was that SVM could separate the data quite efficiently. The best error obtained was 0.1, meaning only 1 of the 10 inputted images was incorrectly classified. The low error rate was promising with regard to the actual data sets. The coefficient used were different in every case, so any correlation between correct classified images and the coefficient used was not yet apparent.

4 Centring Data

4.1 Hypothesis

The retinal images used were often not centred around their maximum, potentially causing the Zernike quantification to have to use a larger number of coefficients in the polynomial to translate the image off centre. The hypothesis was that centring the image before quantification would reduce the number of coefficients used to map the 2D image and therefore the number of basis functions required. The coefficient vector would then be a more accurate representation of the image.

4.2 Preliminary Tests

To check that centring the data did actually reduce the requirements on higher degree coefficients and that this hypothesis was correct, a simulation was run on a set of test data comparing the error using the first 15 coefficients of the quantification vector when the image was centred against when it was in its original state.

	Centred Data Error	Normal Data Error
Data set 1	74.51	70.11
Data set 2	58.30	63.15
Data set 3	74.28	83.02
Data set 4	67.14	88.30
Data set 5	38.03	44.73

Figure 15: Table showing results of preliminary tests

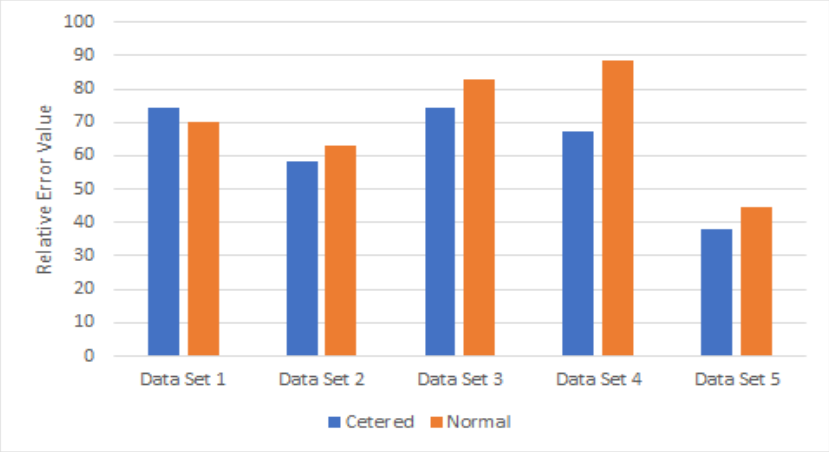

Figure 16: Graphical representation of results visible in figure 15

In four out of five cases above the centred data produced a better error value than the normal data. This supported the initial hypothesis and meaning it was worthwhile performing an investigation as to whether the centring process improved the classification results as well.

4.3 Centring Code Design

To centre any image around its maximum, whilst minimising the data loss occurring from altering the image, a simple method was designed which identified the maximum value of the distribution and then altered the image based on the location of that point.

Pseudo Code:

Input: an original image. (L)
Output: image centred around maximum. (NewIm)

```
#Import data and reshape to size of pre-calculated
#zernike images
let L be image Data
get (x, y) dimensions of L
re-size(L, x/256, y/256)
import ZernikePreComputed
L = L * Zernike{1}

#find maximum of inputed image and position
maxVal = max(L)
(maxx, maxy) = position maxVal

#find closest border of the circle to maxVal
dist = 1
while (not found 0 value)
    #check the value of each position around the
    #max co-ordinates
    if( L(maxx + dist, maxy) == 0)
        dist = dist - 1
    else if( L(maxx, maxy + dist) == 0)
        dist = dist - 1
    else if( L(maxx - dist, maxy) == 0)
        dist = dist - 1
    else if( L(maxx, maxy - dist) == 0)
        dist = dist - 1

#create new image of gathered data
NewIm = l(maxx - dist:maxx + dist, maxy - dist:maxy + dist)

#resize image to the same size as Zerkine images
#and format correctly
(x, y) = size(NewIm)
re-size(NewIm, 256/x, 256/y)
NewIm = NewIm * Zernike{1}
```

4.4 Experiments

To test the hypothesis that centring the images before retrieving their coefficients would improve the classification results, classification experiment 1 was re-run with the same data but this time with centred images.

4.4.1 Experiment

The same 10 images that were used in classification experiment 1 were used again. However, this time, before the quantification of the images was computed they were run through the centring program. The coefficient vectors of the centred images was being used in classification; the value of k and the number of coefficients was altered and the error value recorded. The rest of the experiment remained the same as before.

4.4.2 Results

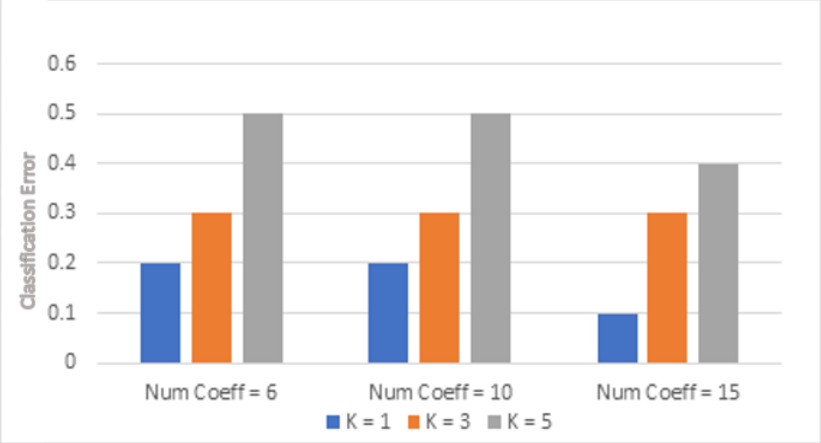

Figure 17: Graphical representation of results obtained in this experiment

4.4.3 Conclusion

Much like in the first experiment, the larger the value of the k the larger the error result. This suggested that a number of outliers were skewing the classification with larger k. However, contrary to experiment 1, the optimal error value arose using 15 coefficients rather than 10, so that for classification of centred images to be effective, higher numbered coefficients are required to provide accurate classification. The optimal outcome in this experiment was using k equals 1 and 15 coefficients producing an error rate of 0.1.

It was inconclusive whether centring the data actually improved the classification, the best error rate being the same in both experiments. Both experiments would need to be run on larger data sets to gain a clearer insight into which is the better option.

5 Real Data Classification Experiments

Having conducted preliminary testing of the algorithms with promising results, the next step was to redesign and re-run the tests on a larger data set. The real data consisted of 106 images to be split proportionately with 70 images being used for training and 36 images used for testing. These were referred to as the training and testing sets respectively. As with the preliminary tests, there were 3 classes of data: 1. Under 50 and healthy; 2. Over 50 and healthy; 3. Over 50 and diseased. Experiments were run using all 3 classes as well as combinations of classes, each experiment begin clearly labelled as to the class or combination of classes tested.

The tests were run as follows:

- Data split into training and testing sets
- For each set of inputs (k, number coefficients, original/centred image) the following data was run
    1. Build a classifier using the training set and the defined inputs
    2. Record the training error
    3. For the top 3 training errors in non centred and centred training results and perform tests on these classifiers using the testing data set.
    4. Record the testing error
    5. Build a confusion matrix from the results, laid out as follows:

Testing Error	Expected Results
Actual Results	Number items classified as such

    6. Produce Receiver Operating Characteristic (ROC) graph to demonstrate specificity and sensitivity of all the confusion matrices. Orange dots to represent the centred results and the blue dots to represent non-centred results.

All of the tests of the KNN-classifier were run and evaluated first, then similar tests were run on the SVM classifier also evaluated. Following completion of the tests an analysis of the results evaluated how the two classifiers performed. Full results for the following experiments are included in the appendix.

5.1 KNN Experiments

5.1.1 Classification into three groups (1 vs 2 vs 3)

The first test was run comparing all three classifications against each other. Being the broadest test, it demonstrated how accurate the classifier was over the whole range. The worst case scenario would have been an error rate of greater that 66% (worse than probabilistically guessing the classification) .

Results:

The results established that the top 3 combinations of values were when the number of coefficients were 15 and the k values 1, 3 and 5. These 3 classifiers were then tested and their testing error and confusion matrices recorded. These values are recorded below.

- K = 1, Number Coefficients = 15

  Coefficients Used: Non-Centred = 4,12,13,15, Centred = 4,6,14,15

Non-Centred:

0.50	1	2	3
1	13	3	4
2	3	2	3
3	2	3	3

Centred:

0.57	1	2	3
1	12	1	3
2	3	4	6
3	3	3	1

- K = 3, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,9,10,11, Centred = 4,8,15

Non-Centred:

0.55	1	2	3
1	12	3	4
2	4	3	5
3	2	2	1

Centred:

0.38	1	2	3
1	17	2	5
2	1	4	4
3	0	2	1

- K = 3, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,4,5,9,10,11,12, Centred = 4,9,10,15

Non-Centred:

0.47	1	2	3
1	12	1	4
2	6	6	5
3	0	1	1

Centred:

0.41	1	2	3
1	16	2	7
2	2	5	3
3	0	1	0

Observations:

The best testing results came with a testing error of 0.3889, representative of 38% misclassified data. However, deeper analysis of the results showed images of the third class were the hardest to classify correctly: the average percentage of misclassified images with classification 3 being 88%. In 2 out of the 3 cases, the centred images produced a better testing error rate than the non centred results. This supports with the hypothesis postulated during the initial centring tests.

Regarding the optimal selection of coefficients, all of the 6 solutions above used one or two of the early coefficients, and then multiples of the later ones. As 5 of the 6 solutions used the fourth basis functions, the results would suggest that the feature this basis function represents is an important factor in separating these classes.

Although the best testing error value of 38% was less than the target 66%, the results were slightly skewed because the majority of the correctly classified values come from images in class 1. Further investigation is required to determine whether this is a problem that recurs in other tests.

5.1.2 Classification into age groups (1 vs 2 & 3)

This test split classification purely by age base. It combined the two classes of over 50 (presumed healthy) and over 50 (unhealthy) to one class of over 50. This was to see if the classifier could successfully split the images by age group. With only two classes the error factor aimed for was less than 50%.

Results:

As in the previous test, the top 3 results for centred and non centred images was again when the number of coefficients was set to 15. These were tested using the testing set and results are shown in the confusion matrices below.

- K = 1, Number Coefficients = 15

Non-Centred:

0.30	1	2&3
1	12	5
2&3	6	13

Centred:

0.36	1	2&3
1	15	10
2&3	3	8

Coefficients Used: Non-Centred = 4,10,12,13,15, Centred = 1,4,7,10,11,14

Non-Centred: Sensitivity = 0.67, Specificity = 0.72

Centred: Sensitivity = 0.83, Specificity = 0.44

- K = 3, Number Coefficients = 15

Non-Centred:

0.30	1	2&3
1	13	6
2&3	5	12

Centred:

0.33	1	2&3
1	14	8
2&3	4	10

Coefficients Used: Non-Centred = 4,13,14,15, Centred = 4,6,12,14,15

Non-Centred: Sensitivity = 0.72, Specificity = 0.67

Centred: Sensitivity = 0.78, Specificity = 0.56

- K = 5, Number Coefficients = 15

Non-Centred:

0.25	1	2&3
1	16	7
2&3	2	11

Centred:

0.22	1	2&3
1	16	6
2&3	2	12

Coefficients Used: Non-Centred = 1,4,9,11,12, Centred = 4,8,11,12,13,14

Non-Centred: Sensitivity = 0.89, Specificity = 0.61

Centred: Sensitivity = 0.89, Specificity = 0.67

Observations:

The best results was a testing error of 22% miss-classification. The issue in 5.1.1where one class was biased in the confusion matrix did not occur in this case as demonstrated by the values of specificity and sensitivity obtained. Miss-classified results were spread more evenly between the two classes. To better visualise these results, the sensitivity and specificity of the top results were plotted on the ROC graph (see figure 18).

Looking at figure 18 one can see that all 3 results for non-centred data were better than 2 of the centred results. It is therefore possible that when looking for a classification between ages, using the original images provides a better split that using centred images.

Examining the coefficients, as before, all of the solutions used one or two of the first early coefficients, and then a number of the later ones. This further supports the observations made in 5.1.1. However, in this experiment the number of later coefficients required to achieve the best results was higher, which suggests that the outlying bumps and divots

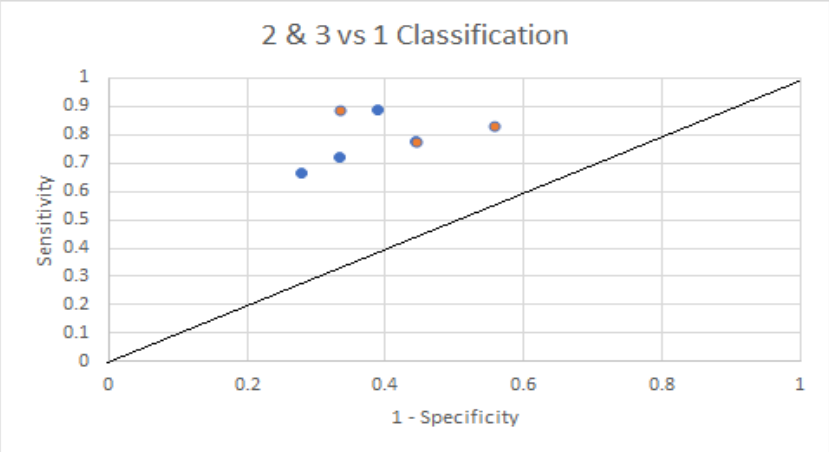

Figure 18: ROC graph of age group experiment

are being used to separate the data require more combinations of the later basis functions. This could indicate that a more specific classifier which looks at more features is required to produce these images.

5.1.3 Classification according to the disease status (1 & 2 vs 3)

This test ignored age and focused on classifying between healthy and unhealthy patients. Therefore, the two healthy classes under and over 50 were combined to form one class of healthy images. As in the previous test an error below 50% was targeted.
Results:

- K = 1, Number Coefficients = 15

Non-Centred:

0.27	1&2	3
1&2	24	8
3	2	2

Centred:

0.38	1&2	3
1&2	21	9
3	5	1

Coefficients Used: Non-Centred = 2,4,5,6,10, Centred = 4,6,8,13

Non-Centred: Sensitivity = 0.92, Specificity = 0.2

Centred: Sensitivity = 0.81, Specificity = 0.1

- K = 3, Number Coefficients = 15

Non-Centred:

0.41	1&2	3
1&2	20	9
3	6	1

Centred:

0.27	1&2	3
1&2	25	9
3	1	1

Coefficients Used: Non-Centred = 2,5,10,11,13, Centred = 4,7,8,9,10,11,15

Non-Centred: Sensitivity = 0.77, Specificity = 0.1

Centred: Sensitivity = 0.96, Specificity = 0.1

- K = 5, Number Coefficients = 15

Non-Centred:

0.25	1&2	3
1&2	25	8
3	1	2

Centred:

0.38	1&2	3
1&2	21	9
3	5	1

Coefficients Used: Non-Centred = 2,4,5,9,10,13, Centred = 7,10,11

Non-Centred: Sensitivity = 0.96, Specificity = 0.2

Centred: Sensitivity = 0.81, Specificity = 0.1

Observations:

The optimal testing error was 27% miss-classified, however, looking at the specificity values of all the top results the best was 0.2. This again represented the bias seen in the first experiment. The combination of a high sensitivity and a low specificity meant that the classifier separated an image of class 1 into the correct class relatively easily, but when looking at an image within class 2, it incorrectly classified it most of the time.

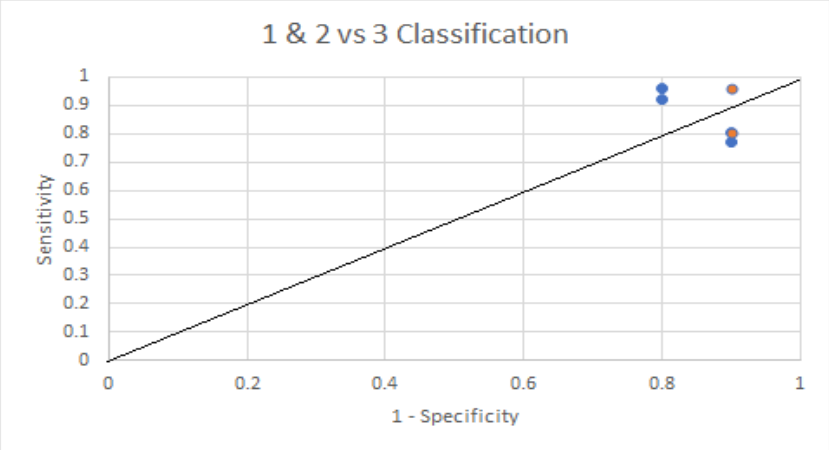

Figure 19: ROC graph of disease status experiment

The graph in figure 19 demonstrates the results visually. The low specificity forced all the plots to be in the top right hand corner. This meant that this classifier was virtually useless, classifying every image as one of the classes. The inaccuracies of this classifier make it inadvisable to use its results to support any claims made with regards to feature selection and therefore looking at the coefficient used would be irrelevant.

5.1.4 Analysis:

Although all the experiments met the criteria set out, in two of the three experiments a bias on the larger data class was clearly visible which skewed the results, something which could be improved to increase the accuracy of the classifiers. The bias arose because the data sets of each class were vastly different in size: 1 being as big as 2 and 3 combined. This theory is supported by the fact that when class 2 & 3 were combined (in experiment 2) the bias was much less visible and the data set sizes were the same.

The bias occurred in the training of the classifier as the when the optimal configuration was chosen it only took into account the number of correctly classified, disregarding class.

This in turn led to cases where all of class 1, 2 or 1&2, is classified correctly but none of class 3 is correct but the overall solution is still the optimal solution. This error was then passed on through to the testing where the problem became visible. A possible solution would be to take the number of data points in a class into consideration when calculating the training error.

In experiment 1 and 2 the observations about which coefficients have been used were the same. The reliance on using one of the earlier coefficients was clearly visible across nearly all the experiments that were conducted. Looking at the extended results in the appendix section, this observation extended to all of the results. This is logical because when looking at the images in their core form, most have the majority of the macular pigment distributed in a central position. This was demonstrated using Zernike basis function 4, with the surrounding humps and divots being represented by basis functions 7 to 15.

5.1.5 Classifier with mean error calculation

To address the issue of the bias present in all of the results in the previous section the classifier was edited slightly. In the training of the KNN-classifier, when the training error was calculated instead of just counting the number of incorrectly classified images it was amended to work out how many in each class were miss classified. It could then work out the fraction of each class miss-classified, calculate the mean of these, and report this as the training error. This removed the bias in all aspects of the classifier. The pseudo code for the change is below, it would be added in at the training error calculation in the original program:

Pseudo Code:

```
if(miss-classified and class is meant to be 1)
    Class1MisClass = Class1MisClass + 1;
else if(miss-classified and class is is meant to be 2)
    Class2MisClass = Class2MisClass + 1;
else if(miss-classified and class is is meant to be 3)
    Class3MisClass = Class3MisClass + 1;

trainingError = ((Class1MisClass/sizeClass1) +
(Class2MisClass/sizeClass2)+(Class3MisClass/sizeClass3))/3;
```

All experiments were now re-run to see if this changed increasing the accuracy of the classifier. Only if the experiment was of a different form to one already completed has any more information than the results and observations been reported. As in all results of the first set of experiments, the top results were consistently achieved when k = 15. Only situations with this value were repeated. So k = 15 was a fixed input for the following tests.

5.1.6 Classification into three groups mean error (1 vs 2 vs 3)

Results:

The confusion matrices representing the results are shown below:

- K = 1, Number Coefficients = 15

  Coefficients Used: Non-Centred = 4,10,13, Centred = 3,4,5,6,9,12,13,15

Non-Centred:

0.47	1	2	3
1	12	2	2
2	4	3	4
3	2	3	4

Centred:

0.36	1	2	3
1	15	2	4
2	2	5	3
3	1	1	3

- K = 3, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,9,11,12,13,14, Centred = 4,9,12,15

Non-Centred:

0.61	1	2	3
1	10	3	6
2	5	4	4
3	3	1	0

Centred:

0.36	1	2	3
1	16	4	5
2	1	4	2
3	1	0	3

- K = 5, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,4,5,9,11,14, Centred = 4,9,10,15

Non-Centred:

0.5	1	2	3
1	12	1	5
2	5	6	5
3	1	1	0

Centred:

0.41	1	2	3
1	16	2	7
2	2	5	3
3	0	1	0

Observations:

Compared with the previous results, changing the error calculation didn't improve the accuracy of the classifier significantly. The best error rate was marginally better at 36%. There was still a problem correctly classifying class 3 (over 50 and diseased) and even in the best case scenario, only 3 out of 10 were correctly identified as diseased.

As in the first experiment for 1 vs 2 vs 3 the centred images provided significantly better results that the non-centred. The fact that this has occurred in both experiments suggests that the features required to separate the classes are more easily detectable when the images are centred than otherwise.

5.1.7 Classification into age groups mean error (1 vs 2 & 3)

Results:

The confusion matrices representing the results are below:

- K = 1, Number Coefficients = 15

Non-Centred:

0.36	1	2&3
1	13	8
2&3	5	10

Centred:

0.36	1	2&3
1	15	10
2&3	3	8

Coefficients Used: Non-Centred = 4,6,10,12,15, Centred = 1,4,7,10,11,14

Non-Centred: Sensitivity = 0.72, Specificity = 0.56

Centred: Sensitivity = 0.83, Specificity = 0.44

- K = 3, Number Coefficients = 15

Non-Centred:

0.27	1	2&3
1	14	6
2&3	4	12

Centred:

0.33	1	2&3
1	14	8
2&3	4	10

Coefficients Used: Non-Centred = 4,9,10,12,13,14,15, Centred = 4,6,12,14,15

Non-Centred: Sensitivity = 0.78, Specificity = 0.67

Centred: Sensitivity = 0.78, Specificity = 0.56

- K = 5, Number Coefficients = 15

Non-Centred:

0.17	1	2&3
1	15	3
2&3	3	15

Centred:

0.22	1	2&3
1	16	6
2&3	2	12

Coefficients Used: Non-Centred = 1,4,5,9,10,11,12,13, Centred = 4,8,11,12,13,14

Non-Centred: Sensitivity = 0.83, Specificity = 0.83

Centred: Sensitivity = 0.89, Specificity = 0.67

Observations:

Changing the error calculation on this classifier improved the results. The best error wass 17% miss-classified. As the bias was not present before, using the mean error was not expected to increase the accuracy significantly, but the change in the sensitivity and specificity values shows that it equalled out the classes. The coefficients used still followed the same pattern that was hypothesised in the first set of experiments, with coefficient 4 being present in every solution. This suggests that classifying between age requires looking at the feature Zernike basis function 4 represents.

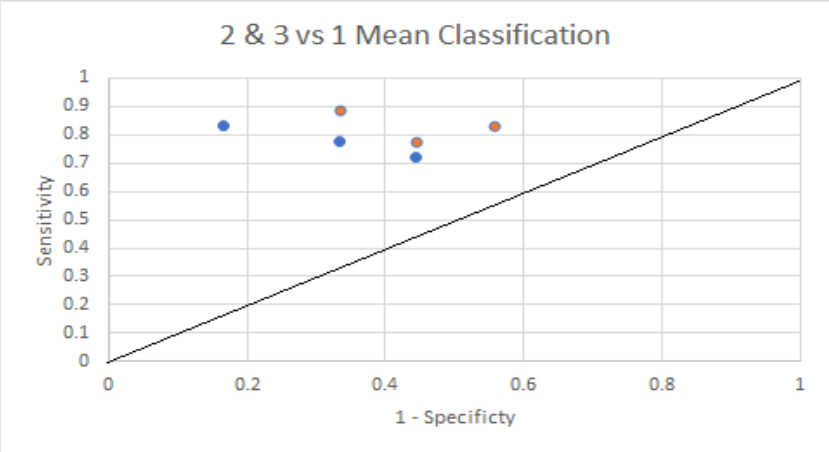

Figure 20: ROC graph of age status mean error experiment

In this case the non-centred results performed better in all instances than the centred results. This may indicate that centring the data alters the features in a decremental way when classifying between ages, as is visible in figure 20. The best result was getting closer to the optimal position: top left corner.

5.1.8 Classification according to the disease status mean error (1 & 2 vs 3)

Results:

The confusion matrices representing the results are below:

- K = 1, Number Coefficients = 15

Non-Centred:

0.39	1&2	3
1&2	23	6
3	3	4

Centred:

0.41	1&2	3
1&2	21	10
3	5	0

Coefficients Used: Non-Centred = 2,9,10,11,12,14, Centred = 1,7,11,15

Non-Centred: Sensitivity = 0.88, Specificity = 0.4

Centred: Sensitivity = 0.81, Specificity = 0

- K = 3, Number Coefficients = 15

Non-Centred:

0.41	1&2	3
1&2	21	9
3	5	1

Centred:

0.27	1&2	3
1&2	25	9
3	1	1

Coefficients Used: Non-Centred = 2,5,10,11,12,13,14,15, Centred = 1,6,7,111,15

Non-Centred: Sensitivity = 0.81, Specificity = 0.1

Centred: Sensitivity = 0.96, Specificity = 0.1

- K = 5, Number Coefficients = 15

Non-Centred:

0.39	1&2	3
1&2	22	10
3	4	0

Centred:

0.36	1&2	3
1&2	23	10
3	3	0

Coefficients Used: Non-Centred = 2,11,13,14,15, Centred = 1

Non-Centred: Sensitivity = 0.84, Specificity = 0

Centred: Sensitivity = 0.88, Specificity = 0

Observations:

The change of error calculation increased the testing error on this classifier. This might be because in the initial tests the bias caused nearly all of class one to be classified correctly. When the classifier was forced to choose the training error best for all classes, the accuracy for class 1 was reduced, thus increasing the error rate.

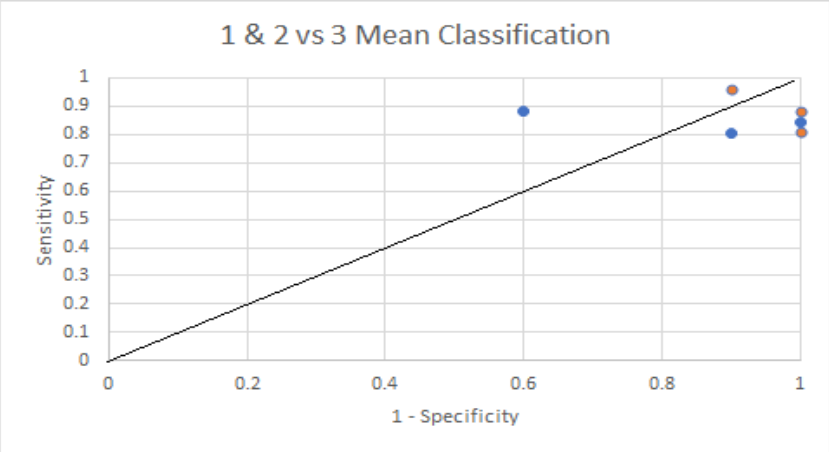

Figure 21: ROC graph of disease status mean error experiment

Figure 21 shows that all of the results are placed in the top right hand corner of the graph. Both the sensitivity and specificity of the classifier was worse with the specificity reaching a value of 0 which means it didn't correctly classify a single image from the second data set.

5.1.9 Classification according to the disease status mean error (2 vs 3)

To try and improve the classifier, this experiment only tested class 2 against class 3, age being removed as a factor completely. It compared solely healthy to unhealthy images of patients over the age 50. It was hoped that this would increase the accuracy, although the data set was only half the size, so the training of the classifier may not have been as effective.

Results:

The confusion matrices representing the results are below:

- K = 1, Number Coefficients = 15

Non-Centred:

0.56	2	3
2	5	7
3	3	3

Centred:

0.67	2	3
2	4	8
3	4	2

Coefficients Used: Non-Centred = 5,6,7,10,11, Centred = 3,4,8,13

Non-Centred: Sensitivity = 0.625, Specificity = 0.3

Centred: Sensitivity = 0.5, Specificity = 0.2

- K = 3, Number Coefficients = 15

Non-Centred:

0.56	2	3
2	7	9
3	1	1

Centred:

0.44	2	3
2	5	5
3	3	5

Coefficients Used: Non-Centred = 5,7,11,15, Centred = 6,8,10,11,15

Non-Centred: Sensitivity = 0.88, Specificity = 0.1

Centred: Sensitivity = 0.63, Specificity = 0.5

- K = 5, Number Coefficients = 15

Non-Centred:

0.56	2	3
2	7	9
3	1	1

Centred:

0.44	2	3
2	5	5
3	3	5

Coefficients Used: Non-Centred = 5,7,10,15, Centred = 11,14,15

Non-Centred: Sensitivity = 0.88, Specificity = 0.1

Centred: Sensitivity = 0.63, Specificity = 0.5

Observations:

The best error recorded was 44% miss-classified, not as successful as previous results. However, the confusion matrix shows that the correctly classified images were split much more evenly than in any other experiment to classify healthy vs unhealthy. This is easily seen in figure 22.

The optimal point was the one point above the line, almost dead centre of the plot area, where the sensitivity and specificity of the classifier was almost equal. Although the error wasn't very low, this set up indicated a promising step towards achieving a better result.

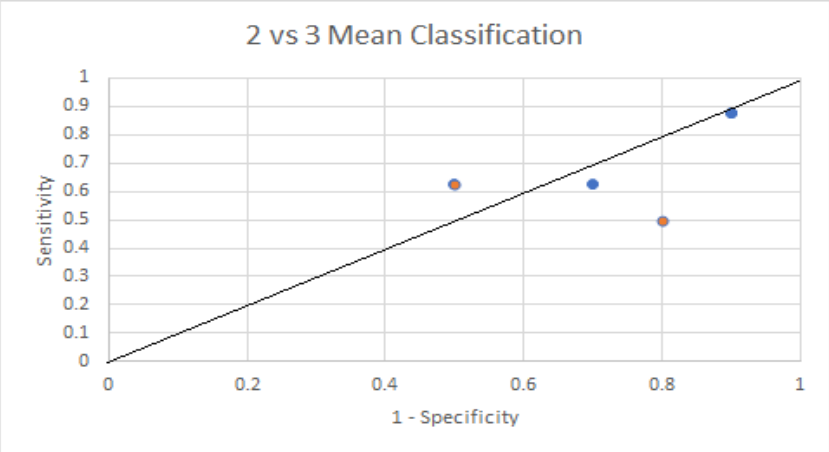

Figure 22: ROC graph of disease status mean error experiment, classes 2 and 3 only

5.1.10 Analysis:

Changing the way the error was calculated had either a negative or positive effect on the classifier depending on which way the data set was split. When the data set was split by age, the accuracy of the classifier increased, but when split by disease the accuracy decreased. The classifier appeared to struggle to split the data into the 3 classes, but was more successful when it split then 2 classes. This may be because different classes require different features to tell them apart and thus different coefficients. However, as previously noted, when the classifier performed well, 2 or 3 of early coefficients, plus a number of the coefficients higher than 10 were required to quantify each of the optimal configurations. As the coefficient 4 was used in almost all of the optimal solutions it would be reasonable to suggest that it is an important factor in splitting the classes.

When designed to decipher the age category an image falls into, the classifier provided a very good result with an error rate of 17%. The coefficients that were used in the optimal configuration were (1,4,5,9,10,11,12,13). It was surprising that such a large number of coefficients were required but this would lead to conclusion that a large number of features were required to distinguish between age groups. It would seem that both the overall magnitude, (coefficients 1,4,5) and the outlying humps and divots (coefficients 9,10,11,12,13) represent important features.

Looking at the actual confusion matrices for both experiments in 1 vs 2 vs 3. Classes one and two were classified fairly accurately but three proved a challenge. A hypothesised solution was to prioritise class 3 over class 1 and 2 which would hopefully mean that 1 and 2 remained fairly accurate whilst 3 also increased in accuracy.

5.1.11 3 Prioritised Classifier:

To test the hypothesis the classifier was changed again slightly. The training will be biased so that when the training error is calculated the percentage of class 3 correctly classified is more important that the other 2 classes. The class 3 percentage has been made 3 times more effective during the training error calculation. Experiments looking at class 1 vs 2 vs 3 and 2 vs 3 have been re run with the new classifier.

5.1.12 Classification into three groups 3 prioritised error (1 vs 2 vs 3)

Results:

The confusion matrices representing the results are below:

- K = 1, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,9,10,11,12,14, Centred = 4,5,6,8,9,11,12,15

Non-Centred:

0.61	1	2	3
1	10	4	4
2	4	3	5
3	4	1	1

Centred:

0.64	1	2	3
1	10	4	7
2	5	3	3
3	3	1	0

- K = 3, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,5,10,11,12,13,14,15, Centred = 1,6,7,11,15

Non-Centred:

0.44	1	2	3
1	12	3	4
2	4	4	2
3	2	1	4

Centred:

0.39	1	2	3
1	17	3	8
2	1	4	1
3	0	1	1

- K = 5, Number Coefficients = 15

  Coefficients Used: Non-Centred = 2,11,13,14,15, Centred = 1,4,6,8,10,111,12,13,15

Non-Centred:

0.5	1	2	3
1	14	3	7
2	1	4	3
3	3	1	0

Centred:

0.5	1	2	3
1	16	6	8
2	1	2	2
3	1	0	0

Observations:

Compared to the results from the other experiments run using all the classes, these results were not as promising as hoped for. The classifier's problem when using all 3 classes was also evident in this experiment. Although the best error rate was 39% it was not representative of the actual accuracy because for class 3 only 1 was correctly classified. Changing the classifier was not successful in this experiment.

5.1.13 Classification according to the disease status 3 prioritised error (2 vs 3)

Results:

The confusion matrices representing the results are below:

- K = 1, Number Coefficients = 15

Non-Centred:

0.61	2	3
1	5	8
2	3	2

Centred:

0.67	2	3
1	4	8
2	4	2

Coefficients Used: Non-Centred = 5,6,7,10,11,13,15, Centred = 3,4,8,13

Non-Centred: Sensitivity = 0.625, Specificity = 0.2

Centred: Sensitivity = 0.5, Specificity = 0.2

- K = 3, Number Coefficients = 15

Non-Centred:

0.33	2	3
1	6	4
2	2	6

Centred:

0.44	2	3
1	6	6
2	2	4

Coefficients Used: Non-Centred = 2,8,11,12,15, Centred = 6,11,15

Non-Centred: Sensitivity = 0.75, Specificity = 0.6

Centred: Sensitivity = 0.75, Specificity = 0.4

- K = 5, Number Coefficients = 15

Non-Centred:

0.44	2	3
1	6	6
2	2	4

Centred:

0.27	2	3
1	6	3
2	2	7

Coefficients Used: Non-Centred = 2,3,13,15, Centred = 9,11,12,15

Non-Centred: Sensitivity = 0.75, Specificity = 0.4

Centred: Sensitivity = 0.75, Specificity = 0.7

Observations:

The best result in this set of experiments was an error rate of 27%, which in this case was quite promising. The specificity and sensitivity shown in figure 23 below indicates that the optimal point moved closer to the top left corner, meaning the miss-classified values were spread fairly evenly across the 2 classes. This is because sensitivity was close to the value of 1 - specificity.

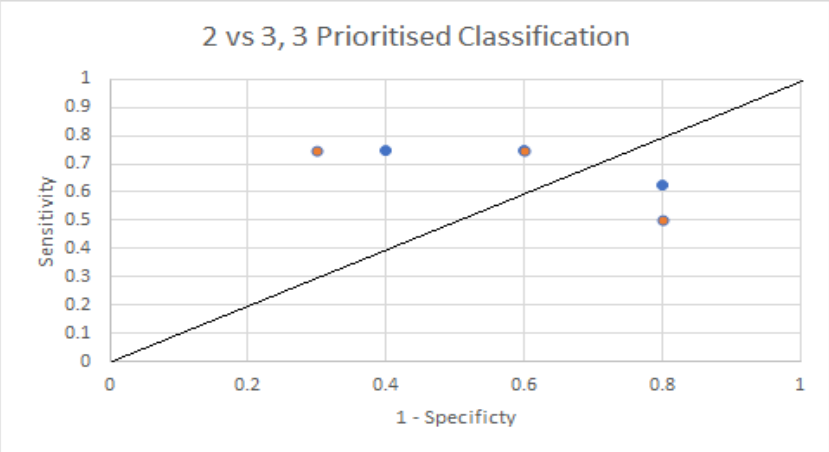

Figure 23: ROC graph of disease status 3 prioritised experiment, classes 2 and 3 only

In this case the optimal solution did not use any of the earlier coefficients and required the image to be centred. It focused on the later coefficients, specifically (9,11,12,15). All of these look at much more at the humps and divots closer to the edge of the image rather than the central cone which could suggest that these play more of a role in identifying whether a patient is diseased or not.

5.1.14 Analysis

In all scenarios where the classifier tried to compare all 3 classes again each other, it produced results with good testing error rates; but which were not representative of the overall accuracy. Every single one had trouble classifying class 3 and even the best results miss-classified 60% of the time. Therefore trying to split between the 3 classes using the KNN-classifier did not work so focusing on methods which split between either age or disease proved a better option.

The classifier produced very good results when splitting the classes purely by age, the best result being only 17% miss-classified. This was when the images were not centred and used coefficients (1,4,5,9,10,11,12,13). The fact that the images did not need to be centred could be because some information is lost in the centring process. This lost data could be the difference between correct and incorrect classification. The coefficients used would support this theory as coefficients 9 to 13 focus on the outlying and very edge sections of the image, so if the image is centred the features at the edges are changed.

The only set up which produced promising results for disease classification was the 3 prioritised experiment. This result was very promising, but as the size of the data set was very small, it would need to be re run to see if it was still accurate. In every solution for this set of experiments the later coefficients were used. This would support the on-going observation that these features, represented by these coefficients, have a big impact on the classification.

5.2 SVM Experiments

As SVM can only classify between two distinct classes the experiments that could be run were limited. Certain classes were combined to try to check for certain conditions. The only user inputted value in these experiments was the number of coefficients that the classifier was allowed to use. This was be limited to 15 when using the powerset operation described earlier. There was also an execution run in each experiment where the classifier was allowed to use 105 coefficients, but where it could not use the powerset operation. This was to see how effective it was when using the full quantification.

5.2.1 Classification according to the age SVM (1 vs 2 & 3)

Results:

The confusion matrices representing the results are below:

- Number Coefficients = 6

Non-Centred:

0.2778	1	2&3
1	14	6
2&3	4	12

Centred:

0.31	1	2&3
1	13	6
2&3	5	12

Coefficients Used: Non-Centred = 2,3,4, Centred = 1,2,4,6

Non-Centred: Sensitivity = 0.78, Specificity = 0.67

Centred: Sensitivity = 0.72, Specificity = 0.67

- Number Coefficients = 10

Non-Centred:

0.31	1	2&3
1	16	9
2&3	2	9

Centred:

0.28	1	2&3
1	14	6
2&3	4	12

Coefficients Used: Non-Centred = 4,7,8,9,10, Centred = 4,5,6,7,8,9

Non-Centred: Sensitivity = 0.89, Specificity = 0.5

Centred: Sensitivity = 0.77, Specificity = 0.67

- Number Coefficients = 15

Non-Centred:

0.19	1	2&3
1	15	4
2&3	3	14

Centred:

0.25	1	2&3
1	17	8
2&3	1	10

Coefficients Used: Non-Centred = 1,3,4,8,9,10,11,13,15, Centred = 1,2,4,5,6,8,9,10,14,15

Non-Centred: Sensitivity = 0.83, Specificity = 0.78

Centred: Sensitivity = 0.94, Specificity = 0.56

- Number Coefficients = 105

Non-Centred:

0.31	1	2&3
1	14	7
2&3	4	11

Centred:

0.39	1	2&3
1	13	9
2&3	5	9

Non-Centred: Sensitivity = 0.78, Specificity = 0.61

Centred: Sensitivity = 0.72, Specificity = 0.5

Observations:

This test aimed to separate the data by the patient's age, and the optimal solution gained was an error of 19% miss-classified. As the number of coefficients increased up to 15 the error decreases on both the centred and non-centred results. However, looking at the results for the the highest number of coefficients, 105 it was the worst result for both experiments meaning that decrease in error must reach a pinnacle at some point between 15 and 105.

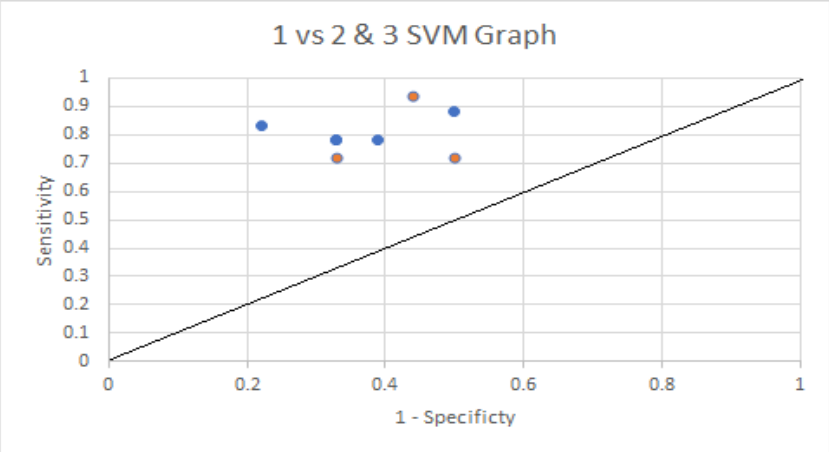

Figure 24: ROC graph of age classification using SVM, testing classes 1 vs 2 and 3

Looking at figure 24 the best result is clearly visible as the one closest to the top left. Both the sensitivity and specificity of this of this result were above 0.75, a very promising statistic. It also means that the miss-classified results were spread fairly evenly. It is clearly visible that 3 of the 4 non-centred results (blue dots) were better than the centred experiments, so centring might not be the best option for this algorithm.

The coefficients used in each showed that in every experiment a large number of the allowed coefficient were used. It is hard to determine the priority of certain coefficients as so many were used, however, this may indicate that all the features of the images were necessary to perform a successful classification.

5.2.2 Classification according to the disease status SVM (2 vs 3)

Results:

The confusion matrices representing the results are below:

- Number Coefficients = 6

Non-Centred:

0.56	2	3
2	5	7
3	3	3

Centred:

0.67	2	3
2	1	5
3	7	5

Coefficients Used: Non-Centred = 2,6, Centred = 2,3,4,8,9,10

Non-Centred: Sensitivity = 0.63, Specificity = 0.3

Centred: Sensitivity = 0.25, Specificity = 0.6

- Number Coefficients = 10

Non-Centred:

0.61	2	3
2	4	7
3	4	3

Centred:

0.56	2	3
2	2	4
3	6	6

Coefficients Used: Non-Centred = 1,2,3,6,7,8,10, Centred = 2,3,4,8,9,10

Non-Centred: Sensitivity = 0.5, Specificity = 0.3

Centred: Sensitivity = 0.25, Specificity = 0.6

- Number Coefficients = 15

Non-Centred:

0.61	2	3
2	4	7
3	4	3

Centred:

0.39	2	3
2	5	4
3	3	6

Coefficients Used: Non-Centred = 4,5,6,7,9,10,11,12,13,14,15,

Centred = 4,5,9,11,12,13,15

Non-Centred: Sensitivity = 0.5, Specificity = 0.3

Centred: Sensitivity = 0.63, Specificity = 0.6

- Number Coefficients = 105

Non-Centred:

0.39	2	3
2	5	4
3	3	6

Centred:

0.5	2	3
2	7	8
3	1	2

Non-Centred: Sensitivity = 0.63, Specificity = 0.6

Centred: Sensitivity = 0.88, Specificity = 0.2

Observations:

The best results occurred at a miss-classification rate of 39% when using non-centred images with 105 coefficients, and centred images with 15 coefficients. The non-centred classifiers performed significantly worse on all occasions when using up to 15 coefficients, but much better than the centred image when using all 105 coefficients. Looking specifically at the 105 coefficient results they could suggest that the good result was due to much more of the image data being available for quantification, compared to the centred image, where information is lost during the centring process. As both of the top results focused on such different features of the image, it is hard to determine any correlation between what allowed the classifier to separate them. However, the result obtained when the number of coefficients was 15 and the image was centred focuses on specific features: (4,5) which showed the overall magnitude of the central peak, and (9,11,12,13,15), which focused on the symmetric qualities of outlying humps and divots.

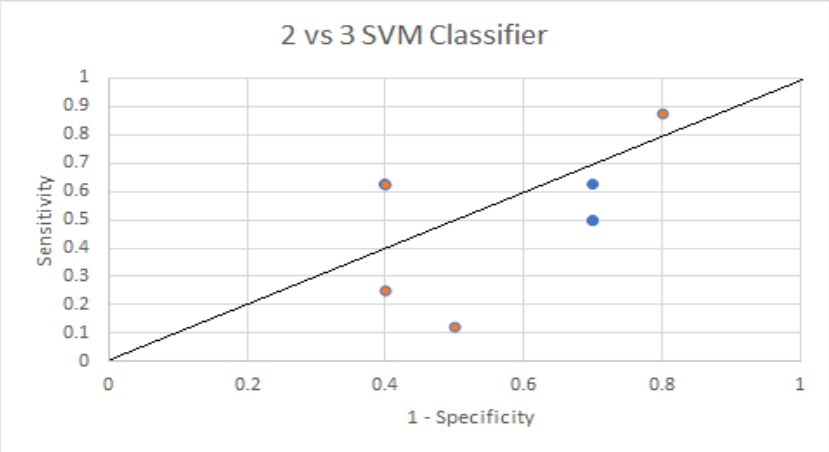

Figure 25: ROC graph of disease status using SVM, classes 2 and 3 onl

Figure 25 re-iterates that the majority of the situations run in this experiment were unusable due to their position below the plotted line on the graph. The specificity and sensitivity of the one result that is usable shows how evenly the miss-classified results were spread, however, the result was only just above the line so it is not as good as it may appear. The one other result that places itself above the decision line is unusable because its specificity is so low, meaning that the majority of this experiments miss-classified images were the diseased ones.

5.3 Comparison of KNN vs SVM

This section is split into 4 parts: classifying between all 3 classes, classifying by age, classifying by disease status, and conclusion comparing the results of each classifier.

The first section, classifying between all classes, was not included in the SVM as it could only take 2 classes into consideration. The KNN is therefore the only choice for this separation. But, as the results obtained in the KNN classifier were not very positive, the earlier conclusion holds that to achieve a more representative result it is best to separate by only 2 classes. Therefore, the results obtained in this set of experiments (1 vs 2 vs 3) were not taken into account when evaluating the classification methods.

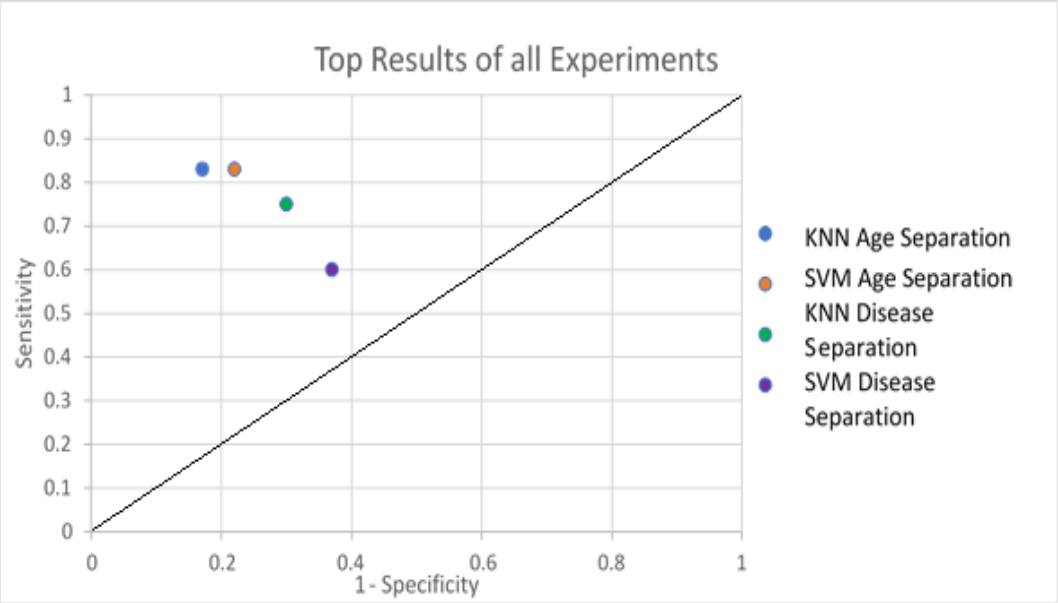

Figure 26: Top results of both methods for age and disease status

When classifying specifically by age, both classifiers performed very well, the best results being 17% for KNN and 19% for SVM. Results were very similar when comparing sensitivity and specificity, both evidencing their miss-classified results evenly spread and allowing the point on the graph to be plotted towards the top left hand corner, see figure 26. Notably, both results were obtained when the image was not centred, so no disparity was evident here either. The number of coefficients used was the same (15), there was only a slight difference in the coefficients that were used. Both followed the pattern that emerged in the KNN test where 2-3 of the early coefficients were used plus a number of the later ones. The only way to differentiate them is that the KNN error was marginally less, setting it slightly above the SVM.

The results when classifying by disease are more separable. The best error for KNN was 28% and for SVM was 39%. Although both errors are considered as good results (below 50%) there is a clear margin of 10% error between the methods. If the tests were run on larger data sets and the error rates stayed true, the difference in the number of miss-classified individuals would be very noticeable. Therefore the KNN was superior in this experiment.

Although both classification methods were were similar in the second type of classification, the KNN produced the optimal solution in both and was vastly superior in the third separation. Therefore, it would be the optimal choice to use and it would be useful to investigate further possibilities of the KNN classification method.

6 Evaluation of the success of the Quantification and the Classification

6.1 Results of the Project

The first part of the project, which was the quantification, achieved its goal as using the Zernike polynomials provided an accurate representation of any image. Any image that was inputted into the optimisation tool created could be quantified, and the tool successfully outputs a coefficient vector that was representative of the image. Any one of the vectors produced could be used to reconstruct an accurate representation of the image, justifying the success of the quantification.

The classification tool demonstrated that it could split the data by age with an error rate of only 17%, and by disease status with an error rate of 28%, but when it split by both at the same time, the best error rate was 36%. The worst error rate of 36% was also not representative of the true nature of the results because it had high sensitivity and very low specificity. Splitting by both conditions did not provide good results which led the research to focus on combined classes. In both cases the tool used specific features of the images represented by different Zernike basis functions to split the data. The features focused on in age classification were the basis functions at positions: 4,11,13. These appeared more than any others and are shown individually in figure 27.

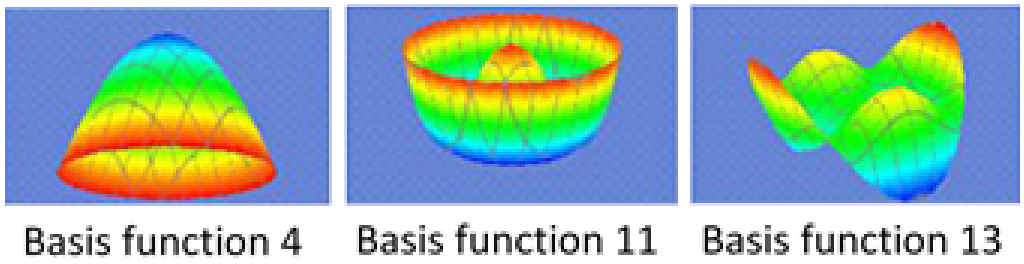

Figure 27: Most common basis functions used for age classification

These three plots show three of the features or the distribution of pigment that is used to classify the data by age. The fact that the classification error rate is directly linked to the features of the distribution, and that it is so low, would suggest there is relatively strong correlation between the distribution of these features in an image and how old the patient is. Further experiments would need to be conducted to see if the results held up in another but this initial set of results were very promising.

Similar conclusions can be drawn for splitting between the healthiness classifier. However, the correlation is slightly weaker as the error is greater and many more extensive experiments were required to get to an efficient classifier. Another difference is reflected in the features that the correlation is present in. In this case it is Zernike basis functions 9, 11, 12 and 13, see figure 28.

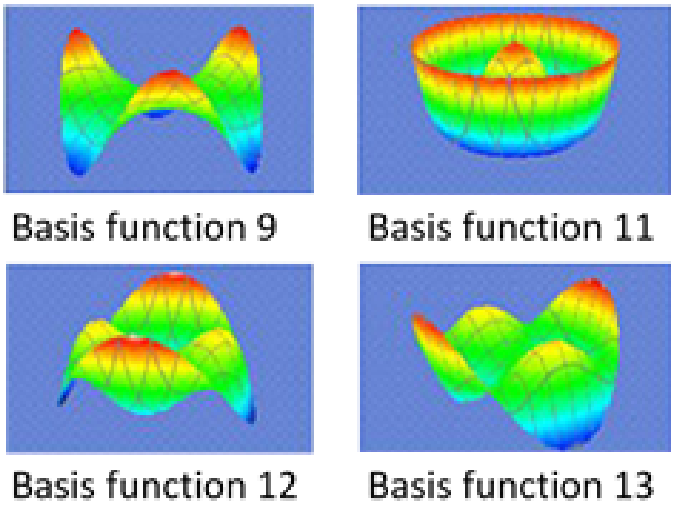

Figure 28: Most common basis functions used for disease status classification

Three of these features represent the surrounding peaks and divots of the central pillar whilst one looks at both the central pillar and the surrounding features. The top solution required the image to be centred first, reducing the distortion of the later coefficients that comes with having to translate the central peak. This means the coefficients were more representative of the actual features required for that classification.

Considering how accurately the Zernike polynomials quantified the images and the success of the classification techniques the research would suggest that using Zernike polynomials for this problem proved to be a very good solution.

6.2 Further Development

To increase the accuracy of the classifier and the reliability of the results it is suggested that a second experiment is carried out with a new and considerably larger data set. The training of the classifier could be improved by using a larger amount of data. Another area of investigation is the KNN-classifier. The 3 prioritise method produced the best result, however, how much priority you give class 3 and whether you change the bias to the other classes as well to see how that changes the results. This research looked at two different classifiers, however, there are many more that might be more suitable for high dimensional data inputs. These, if explored, could provide better results than the ones that have already been obtained. Possible classifiers to look at are Quadratic Discriminant Analysis, Neural Networks and Decision Tree Learning.

One of the main issues was that during the classifier's training state the powerset operator is used which has a complexity of 2^n, so the number of combinations of coefficients that can be checked is limited by computing power. Instead of using the powerset another possibility might be to use dimensional reduction techniques, like principal component analysis, to reduce the number of dimensions before the powerset computation is carried out.

7 Conclusion

The earlier work described in the background information was not able to classify the images using the Gaussian solution. However, the new quantification technique using Zernike Polynomials explored in this project performed very well, accurately quantifying any images to an error rate of up 10^{-12} which is negligible. The Classification techniques identified correlations between the distribution of macular pigment in the fovea and a patient's age or healthiness, but not both at the same time. A number of methods of optimisation and classification were tested thoroughly and informed decisions were made to choose the best methods for this project. Overall the goal of the project was completed, and has created more possible lines of research to increase the accuracy of the results and possibly, in the future, provide diagnostic information to doctors on a patient's health.

8 Project Management & Critical Analysis

In the project proposal, an initial outline of the how much time would be spent on each area was laid out. During the project this outline was held to well and no one section took longer than initially estimated.

The only issue that arose was that initially only one classifier was going to be investigated, however it became clear that it would be appropriate to compare results. This added time pressure at the end of the project but a second classifier was investigated and the time line was still met.

Included below is an outline of the actual time-line of the project, on a week by week basis.

- Week 1: Write project proposal & initial set of project resources
- Week 2 – 4: Learn Matlab syntax, Initial experiments into Zernike Polynomial manipulation, gain understanding of project requirements.
- Week 5: Build Zernike calculation function
- Week 6 – 12: Research and code optimisation algorithms
  - Week 6 -7: research techniques and shortlist possible methods
  - Week 8-10: Implement shortlisted method in Matlab
  - Week 11-12: Perform algorithm testing on implemented methods
- Christmas Break: Clean up optimisation code and start writing up report section on parts of projects already completed
- Week 13-18: Research and code classification algorithm
  - Week 13: research possible methods
  - Week 14-15: Implement chosen method
  - Week 16: build evaluation framework
  - Week 17-18: Evaluate classification technique
- Week 19-20: Add in second classification technique and evaluate.
- Week 21 onwards: Investigation into future approaches.

9 References

1. Styles, I.B., Calcagni, A., Claridge, E., Orihuela-Espina, F. and Gibson, J.M.. Multispectral retinal image analysis: a novel non-invasive tool for retinal imaging. www.nature.com/eye, 7, 201

2. Howells, Olivia, Frank Eperjesi, and Hannah Bartlett. "Improving the repeatability of heterochromatic flicker photometry for measurement of macular pigment optical density." Graefe's archive for clinical and experimental ophthalmology 251.3 (2013): 871-880.

3. Macular Pigment, 2017. [ONLINE] Address: http: //www.eyepromise.com/doctors/about/macular-pigment/

4. Zernike, F. 1934. "Beugungstheorie des Schneidenverfahrens und Seiner Verbesserten Form, der Phasenkontrastmethode". Physica. 1 (8): p.689–704.

5. Eric C. Kintner, 1976. "On the Mathematical Properties of the Zernike Polynomials". Optica Acta, vol. 23 (8): p.679-680.

6. Wikipedia. 2016. Zernike Polynomials. [ONLINE]

   Address: https: //en.wikipedia.org/wiki/Zernike_Polynomials

7. Elena Deza, Michel Marie, 2009. "Encyclopedia of Distances". Springer p.94

8. Gene Golub; Charles F. Van Loan, 1996. Matrix Computations – Third Edition. Baltimore: The Johns Hopkins University Press. p. 53

9. Rob Haelterma, 2009. "Analytical study of the least squares quasi-Newton method for interaction problems". PhD Thesis, Ghent University. Retrieved 2014-08-14.

10. R Fletcher, 1970. "A New Approach to Variable Metric Algorithms", Computer Journal, 13 (3): p.317–322

11. C G Broyden, 1970. "The convergence of a class of double-rank minimization algorithms", Journal of the Institute of Mathematics and Its Applications, 6: p76–90

12. D Goldfarb, 1970. "A Family of Variable Metric Updates Derived by Variational Means", Mathematics of Computation, 24 (109): p23–26

13. David F Shanno, 1970, "Conditioning of quasi-Newton methods for function minimization", Mathematics of Computation, 24 (111): p647–656

14. Wikipedia. 2016. Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. [ONLINE]

    Address: https: //en.wikipedia.org/wiki/Broyden–Fletcher–Goldfarb–Shanno_algorithm

15. Kenneth Levenberg, 1944. "A Method for the Solution of Certain Non-Linear Problems in Least Squares". Quarterly of Applied Mathematics. 2: p164–168

16. Donald Marquardt, 1963. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". SIAM Journal on Applied Mathematics. 11 (2): p431–441

17. Madsen et al, 2004. Methods for Non-Linear Least Squares Problems (2nd ed.)

18. IB Styles, 2016. Computational Tools for Modelling and Analysis: Fitting Models to Data. p1-3

19. Michal Vose, 1999. The Simple Genetic Algorithm: Foundations and Theory. Cambridge, MA: MIT Press.

20. Miller, Brad; Goldberg, David, 1995. "Genetic Algorithms, Tournament Selection, and the Effects of Noise". Complex Systems. 9: 193–212.

21. Baker, James E, 1987. "Reducing Bias and Inefficiency in the Selection Algorithm". Proceedings of the Second International Conference on Genetic Algorithms and their Application. Hillsdale, New Jersey: L. Erlbaum Associates: p14–21

22. Marek Obitko, 2011 "XI. Crossover and Mutation". http://www.obitko.com/:

23. Tomasz D. Gwiazda, 2006. Genetic Algorithms Reference Vol.1 Crossover for single-objective numerical optimization problems, Lomianki.

24. Altman, N. S., 1992. "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): p175–185

25. Cristianini, Nello; and Shawe-Taylor, John, 2000; "An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press.

10 Appendix

10.1 Structure of attached Zip file:

The zip file contains the following:

- Source Code folder.
- Training and Testing Data: This holds all the training and testing data required to re-run the experiments completed within the project.
- Read Me: This goes through an in depth explanation of how to run the source code.
- Extended appendix: This holds all of the tables of extra data.
- Final Year Report: Pdf of the report

10.2 Running the Source Code

This section will explain how to run the different sections of source code.

To create the library just run ZernikePreCompute(); in Matlab. The 'dim' value can be altered to create a dictionary of different sized images.

```
1   function ZernikePreCompute()
8 -   dim = 256;
```

All optimisation algorithms are run the same way. They take in one input of an image and return the error and the coefficient vector.

```
1  Image = 'the image data';
2  [coefficient vector, error] = SolverLinearSquare(image);
3  [coefficient vector, error] = SolverLevenbergMarq(image);
4  [coefficient vector, error] = SolverQuasiNewton(image);
5  [coefficient vector, error] = SolverGeneticAlgorithm(image);
```

For KNN the user inputs a value of k, numcoeffs, and the data being for training into the KNN training method chosen. The SVM only uses the numcoeff Input. Eval is the evaluation matrix holding the information on how the classifier performed.

```
1  testData = 'Test data set'; trainData = 'Training data set';
2  k = 'k value to use'; numCoeff = 'num coeffs allowed to use'
3  %for KNN methods trainings and testing
4  KNN_Class_Build(k, numCoeff, trainData);
5  [Eval] = Knn_Class_Test_Classify(testData);
6  KNN_Class_Build_Average(k, numCoeff, trainData);
7  [Eval] = Knn_Class_Test_Classify(testData);
8  KNN_Class_Build_3Prioritised(k, numCoeff, trainData);
9  [Eval] = Knn_Class_Test_Classify(testData);
10 %for SVM
11 SVM_Classifier_Train(numcoeff, trainData);
12 [Eval] = SVM_Classifier_Test(testData);
```