Medical Image Processing - Lesions

Author: Corey White
Supervisor: Paul L Rosin
Moderator: Alia I Abdelmoty
CM3203 - 40 credits

Table of Contents

Part I

Abstract

Melanoma is an ever growing threat especially for Caucasian individuals, with the incident rates increasing, there may come a time where there may be too many patients to diagnoses effectively with the traditional methods causing melanoma to go untreated for longer. As melanoma can be fatal in later stages, but cured in earlier stages it is of vital importance that a quick and efficient diagnosis is made.

This project aims to offer an alternative method of diagnosing Melanoma by employing image processing techniques to emulate the diagnostic process. Features are extracted from the lesion initially following the popular ABCD mnemonic. Once these features have been extracted they are used to train a classifier, in his case a Support Vector Machine. The trained classifier is then used to distinguish between the different types of skin lesion. As malignant melanoma can be terminal especially in the later stages, this project shall be evaluated on its ability to correctly classify malignant skin lesions.

Part II

Acknowledgements

- Thanks to Paul Rosin, you have been an excellent supervisor helping with any questions that I had.
- Special thanks to my family, during this time of personal loss and bereavement, you have all supported me both emotionally and through my work, I would not have been able to do this without it.
- Thanks to Rebecca Wigfall, listening to all my ramblings and private lectures when most may have fallen asleep you always remained interested.

Part III

Body

Chapter 1
Introduction

According to cancer research UK cases of malignant melanoma have risen faster than any other leading cancer in the UK over the last thirty years. In fact rates have increased over more than fivefold compared to incidence rates in the 1970s. [1]

In 1992 it was concluded by the International Agency for the Research on Cancer, that there was "Sufficient evidence for the carcinogenicity of solar radiation", categorising ultra violet light as a group one carcinogen. This category is reserved only for agents or mixtures that are known and proven to be hazardous to humans, other agents grouped into this category consist of Asbestos and Neutron radiation.

Unfortunately there has been an increase in ultraviolet radiation over the last thirty years due to global warming, this seems to correlate with the increase in malignant melanoma. As global warming is an increasing phenomena and the amount of ultraviolet radiation will consistently increase, as will the risk of melanoma especially in agricultural and other employment that increases one's exposure to sunlight.

Fortunately the survival rate for melanoma has increased significantly over the last forty years. Survival rates are the standardized way of discussing the prognosis of a patient, the five and ten year survival rates indicate the percentage of patients that survive at least this long after diagnosis (although it must be noted that a lot of patients survive much longer and in many cases the melanoma can be cured). [2]

Below is a table showing the survival rates of each stage of melanoma.[2]

Stage	5-year survival rate (%)	10-year survival rate (%)
Stage IA	97	95
Stage IB	92	86
Stage IIA	81	67
Stage IIB	70	57
Stage IIC	53	40
Stage IIIA	78	68
Stage IIIB	59	43
Stage IIIC	40	24
Stage IV	15 - 20	10 - 15

You may notice that Stage IIC has a lower survival rate than that of Stages IIIA and B, this is due to the ulceration on the skin surface that increases the risk of the cancer spreading, and such ulceration is less likely in the later stages mentioned.

Aside from the anomaly previously mentioned you can clearly see a decline in survival rates as melanoma continues to go untreated, and with the incident rate steadily increasing it is

now more important than ever to diagnose melanoma in its earliest possible stage to decrease the mortality rate of such a curable form of cancer.

Due to the physical time constraints of doctors, there is only a specific number of patients that specialists can see per a year, as the amount of melanoma cases are increasing we may soon reach a point where the number of melanoma cases will potentially outweigh the number of patients a physician can see per a year. Therefore it is of increasing importance that alternate diagnostic tools be created that would allow for a faster, more efficient diagnosis reducing the amount of time required for each patient, and therefore increasing amount of patients diagnosed at earlier stages and thus decrease the overall mortality rate of melanoma.

The purpose of this project is to fulfil this need by emulating the diagnostic process of the early detection of melanoma by examining the physical characteristics of the skin lesion. This project was initially inspired by a paper published by Ganster et al [9] which also attempted categorise and detect malignant skin lesions, the main goal would be to achieve similar if not superior results than this paper.

The data set I have been provided with consists of four types of data, “Non-Melanocytic” Skin Lesions, “Benign” skin lesions, “Dysplastic” skin lesions and “Malignant” skin lesions. Due to the small number of Malignant and Dysplastic skin lesions in the data set, as well as their common characteristics, I shall be combining the two groups into one throughout this project. I shall also be combining the “Non-Melanocytic” Skin Lesions and “Benign” skin lesions into a separate group.

Chapter 2
Background

Using image processing to categorise melanoma skin lesions is certainly not a new idea, there have been many papers describing different techniques of how to solve this particular problem. In fact as mentioned this project was initially inspired by the works of Ganster at al [9].

Source	Segmentation Method	Classifier	Number of Images	% of Malignant	% of Dysplastic	Sensitivity	Specificity
[14]	Thresholding	NR	246	26	45	100	84
[9]	Thresholding & Colour Clustering	kNN	5363	2	19	73	89
[15]	Thresholding	ANN	58	38	19	77	75
[16]	Edge Detection	ANN	147	39	29	93	92.75
[17]	None	CART	40	50	30	100	91
[18]	Thresholding & Region Growing	ANN	319	24	59	86.6	90.2
[19]	NR	Logical Regression	837	10	11	80.0	82.4
[20]	Semi-automatic & manual	Logical Regression	2430	16	25	91	65

The first challenge that most of the methods detailed above have encountered is the segmentation of the skin lesion from its surrounding skin, this would allow for a local analysis of the lesion itself without any anomalous data from the skin skewing the results. The easiest method by which one could segment the lesion with a high level of results is making the segmentation process fully or at least semi manual as in [20]. This however would be very time consuming and contradict with one of this project's key objectives, making the system fully automatic.

Another method that has been used previously for the segmentation process is edge detection, this is the process of detecting the boundaries of an object by detecting the sharp changes in brightness between the desired object and the background. There are many methods of edge detection that can be used, generally edge detection algorithms are categorised into two classes gradient based edge detectors ad Laplacian edge detectors. The Laplacian method has been used by [16] in their attempt at the project, Laplacian works by searching for the zero-crossings in the second derivative of the image.

The most popular technique used in previous attempts is thresholding, its popularity is likely due to its simplistic nature yet generally accurate results. In a majority of the images provided, the pixel values for the (grayscale) lesion are substantially different from that of the surrounding skin, thresholding methods can be categorised into two classes, global thresholding and local thresholding.

In global thresholding a single thresholding value is used over the whole image to dichotomize the pixels of the image into two classes, local thresholding the image is split into smaller segments an thresholding value is found for each segment and is used to dichotomize the smaller segments as in global thresholding, the results are then recombined into a single image.

Because of its innate simplicity, global thresholding has been a popular method for many years and has been widely used in the classification of skin lesions [9, 14, 15, and 18]. Fortunately over the years several successful techniques have been proposed using a variety of different approaches including an analysis of an intensity histogram such as the Otsu [5] and Kittler and Illingworth [21] methods.

One of the main guides by which many papers attempting to solve this problem is the ABCD mnemonic, this mnemonic is currently still used in diagnostics today to diagnose melanoma.

Before Friedman et al published the ABCD mnemonic in 1985 [6] detection of melanoma was mainly achieved by macroscopic features such as bleeding and ulceration. Unfortunately however these features are mostly found in advanced cases where curative surgery may not be an option, and therefore diagnosis through different characteristics such as the ABCD(E) mnemonic became more popular and remains a cornerstone in early melanoma diagnostics today.

In the mnemonic A stands for Asymmetry, unlike benign skin lesions which tend to be round in shape and symmetric, dysplastic or malignant skin lesions are usually irregular in shape and asymmetric. [6]

B stands for border irregularity, unlike benign skin lesions which tend to have smooth, regular borders, the borders of dysplastic and malignant skin lesions are predominantly rough, misshapen and fractal in nature.[6]

C stands for colour variation, dysplastic and malignant skin lesions tend to be much more variegated, displaying different hues of brown or black and in a few cases red or white colours, colours in benign lesions are usually uniform throughout the lesion. [6]

D stands for diameter, benign skin lesions are predominantly less than six millimetres whereas dysplastic and malignant skin lesions are generally larger than six millimetres. The collection of skin lesion images is the same library of images used by Ganster et al, it is unknown whether these images were all taken at a uniform distance from the subject and if this is not the case diameter cannot be obtained. However it can be inferred that the distance is uniform, as Ganster et al used the area of the lesion as a feature calculation which would otherwise give inconclusive results. [6]

E stands for evolution [7], this is the analysis of the same skin lesion over a period of time measuring any significant changes in its appearance. Unfortunately while this would be a useful and effective method in the detection process, I do not have images of the same skin lesion over a period of time and therefore will not be able to emulate this portion of the diagnostic process.

While these are good features to extract from the lesion there have been other papers that decided not to use the ABCD mnemonic as a basis for their classification, instead using vision algebra algorithms [19]. They applied a large number of algorithms of vision algebra to the pictures, as reported in the standard textbook on vision algebra by Ritter and Wilson [13]. These analytical parameters included morphological parameters, geometric parameters such as surface area and extent, invariant moments, symmetry and colour analysis. [19]

Once I have extracted a sufficient number of features from the lesion image, I will employ machine learning to create a categorisation process. Given the large amount of data and the

non-evolving nature of the data in question I have chosen to implement the Support Vector Machine (SVM) classifier. I will experiment with different amounts of training data, increasing the amount of training data until results begin to deteriorate.

The final challenge is using a training a classifier with the extracted features. This trained classifier is what shall be used to categorise the list of skin lesions. In previous attempts at this project multiple classifiers have been used, such as K-Nearest Neighbour [9], logistical regression [19] and Artificial Neural Network [15].

Chapter 3
Methods and Algorithms

In this chapter I shall give a detailed explanation of each algorithm that has been used in this project, showing the equations that have been used and giving worked examples where I feel it is necessary. The main goal of this chapter is to not only display my understanding of each algorithm but also assure that, you, as the reader also fully understand all algorithms and methods used.

Segmentation

The first challenge of this project was to create an algorithm capable of automatically separating the skin lesion on the provided ELM images from the surrounding skin. The skin lesions found on the images provided display large variations in size, shape, position, colour and contrasts, therefore the resulting algorithm must be capable of adapting to the environment of each individual image. Each image may also be accompanied by a modicum amount of noise, this may be due to additional structures on the skin's surface aside from the lesion, this could potentially include damage to the skin, ulceration upon the lesion itself [6], but the most common form of noise witnessed on the images provided was body hair. As hair appears as long dark structures in the image and may be large in number, they could potentially cause issues for segmentation methods that rely on the light intensities of a grayscale image such as threshholding.

To overcome this potential problem, I decided to perform a small amount of pre-segmentation processing, I opted to blur the image before performing any segmentation, blurring would reduce the amount of noise in the image and would allow the lesion to be found more easily. The downside of using this blurring method may be a loss in accuracy with lesions that have similar skin pigmentation to that of the surrounding skin, it became a question of which would produce the greater gain. Through several iterations I found that by using the blurring method it provided me with a greater accuracy over a larger amount of images,

Global Segmentation

Thresholding

Otsu

The Otsu thresholding method is named after Nobuyuki Otsu, who initially described the method in his 1979 publication [5]. The method automatically performs cluster based image segmentation, reducing a grayscale image into a logical binary image that shall be used as a mask. The Otsu thresholding method only requires the histogram of the pixel intensities and no other prior knowledge, this is advantageous as an essential requirement of this project is automatic detection of skin lesions without any required human interaction. The objective behind this method is to find the

thresholding value that maximizes the between class variance or minimizes the within class variance, as the between class variance is simpler to calculate I shall use that in my algorithm.

The first step in the Otsu thresholding method is to create a histogram of all the possible intensity values (0 – 255), the histogram indicates the number of times that each intensity level has occurred in the given grayscale image. The histogram is generated by creating an array whose size is equal to the number of possible intensities, each time an intensity is encountered in the given image its corresponding element in the array is incremented. [5]

For example, let us assume that the number of possible intensity values in an image is represented by L, the number of pixels with the given intensity level i (i = [1,2,3….L]) is then represented by n_i such that the sum of all n_i is equal to the number of pixels in the given image, which we shall call N. [5]

$$\sum_{i=1}^{N} n_i = N$$

The next step would be to calculate the probability of each intensity occurring in the image, the probability of each intensity i occurring in the given image shall be represented byp_i, p_i is calculated by dividing the each n_i by the total number of pixels N. The sum of the probability distribution should be equal to 1.

$$p_i = \frac{n_i}{N} \sum_{i=1}^{256} p_i = 1$$

Now each intensity level is considered as a thresholding level k, and at each thresholding level the pixels are dichotomized into two classes, a background class which would represent the surrounding skin and a foreground class that would represent the skin lesion we would like to find.
For each value of k, the weight of both classes is calculated, the foreground class weight is calculated as the sum of all intensity probabilities from 1 to the thresholding level k. The background class weight can be calculated as the sum of all intensity probabilities for all thresholding values above k. [5]

$$w_f = \sum_{i=1}^{k} p_i \, w_f = \sum_{i=k+1}^{L} p_i$$

The next step is to calculate the foreground and background class mean represented as $u_f$and u_b respectively. The value of u_f is calculated as the sum of each intensity probability multiplied by its corresponding intensity, all divided by w_f for all values up to the thersholding limit k. The value of u_b is calculated as the sum of each intensity probability multiplied by its corresponding intensity, all divided by w_b for all values above k.

$$u_f = \sum_{i=1}^{k} \frac{ip_i}{w_f} u_b = \sum_{i=k+1}^{256} \frac{ip_i}{w_b}$$

As I have stated previously the optimum thresholding value is obtained by finding the intensity that has the maximum between class variance or the minimum within class variance in my algorithm I decided to entirely focus on the between class variance as it is the simplest to calculate. The between class is calculated by multiplying the product of both class' weights with the background class mean minus the foreground class mean squared. [5]

$$\sigma^2{}_b = w_f w_b (u_b - u_f)^2$$

Local Thresholding

Up until now, we have been performing segmentation on a global level, as has been discussed, while this process is generally affective there are certain limitations. As this segmentation process will be automatic with no human interaction, it will be desirable to overcome these limitations as best we can, to do this I shall consider the local segmentation method "Dynamic Thresholding".

Dynamic Thresholding

Dynamic thresholding works by simply splitting the provided mage into smaller segments, once these smaller segments have been generated a thresholding method (in our case the Otsu method) is performed on each one, the results are then combined into a single logical binary mask. This will potentially overcome global thresholding's limitations as the portions of the given image with different levels of intensities may be separated into a different section and therefore not adversely affect one another. Dynamic Thresholding also comes with its own drawback that will need to be considered, a potential increase in noise is.

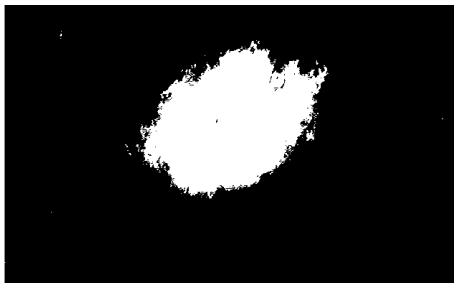
The first challenge of dynamic thresholding is deciding how to separate the image into segments, to resolve this issue I decided to calculate all divisors of the height and width on the image that are greater than or equal to ten percent.
For each of these divisors found I split the image into strips along its respective dimension and performed the Otsu thresholding method on each strip. I then combined the segmented strips into a single logical binary image which would be mask or that divisor. The result is several masks which will then be stored for the fusion process later.

Colour quantisation

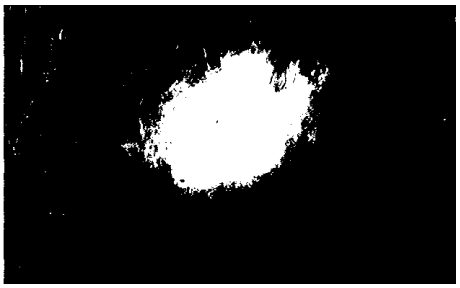
The next method of image segmentation I have implemented is rather different from the segmentation methods I have implemented previously, this method is called Colour Quantisation [22]. Colour Quantisation is a process that reduces the total number of unique colours in an image, an average RGB image has a potential 16,777,216 possible unique colours, a majority of these colours will most likely not occur in most images and a percentage of the colours that do occur in an image will likely have different values but be imperceptibly different from other colours in the image. Therefore it would be desirable to use this method so that the imperceptibly different colours to be represented by the same colour, therefore reducing the total number of colours in the image and reducing the size of the file.

Colour Quantisation can be thought of an extension of vector quantisation, simply stated vector quantisation is the selection of I vectors in some N dimensional space, in order to represent M number of vectors that reside in that same dimensional space. The result is the sum of the collection of vectors I being smaller than the collection of vectors M and the total errors incurred by the process being minimized. Colour quantisation can therefore be thought of as a process of vector quantisation in three dimensional space (i.e. RGB, HSV, Lab etc.).
Colour quantisation can be broken into four phases, phase one is responsible for sampling the original image's colour statistics, phase two takes the colour statistics from phase one and using them to create a colour map, phase three maps the colours in the original image to the colour map created in phase two, finally phase four is responsible for quantising and drawing the resulting image.

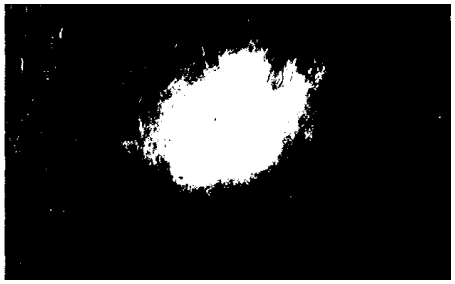
Instead of using Colour quantisation in the traditional manner, i.e. creating a colour map of 256 colours to represent all colours in an image, I have decided to use colour quantisation to find and map only two colours, one for the surrounding skin and another for the lesion itself, I then convert the resulting image into a logical binary image, this will be the mask for that image.



Colour Quantised RGB Image



Colour Quantised Lab Image



Colour Quantised CbCr Image

Mask Fusion Process

Once I have gathered all the masks from the segmentation methods described above, I begin the process of fusing them into a single final mask. This achieved by a process called

"Majority Voting", I add all the generated masks together , this results in a single matrix whose each element equals a vote for its corresponding pixel. If more than half of the masks vote for a pixel (i.e. its value in the generated matrix is higher than half the total number of masks) then its value is set to 1, if less than half of the masks vote for a pixel, then that value is set to 0. The results is a single binary mask, however there may still be residual noise outside of the segmented lesion, so some additional processing is applied to the mask to rid it of this noise.

Connective component labelling

The process I have decided to use is called "Connective Component Labelling" [3]. Simply stated, connective component labelling is a process of searching through a matrix for a specific value (in this case the number 1), each separate group is then labelled by a different value.

Once I have labelled the separate groups of 1s in the matrix, I apply a series of rules to ascertain which groups I will discard and which shall be kept.

1) Any group growing into the edge of the matrix is discarded.

2) The smallest groups are discarded leaving only the largest group

3) Any holes in the last remaining group is filled.

The rules I have applied above generally give good results, there are a few cases however in which these rules fail. The main cause of failure are the few lesions that do grow into the edges of the image. Generally when this issue presents itself, it gives an empty or very small mask, if ever this is the case, the mask with the largest area is then used.

Asymmetry

In their publication, Ganster et al [9] stated the importance of the analysis of symmetry for a lesion, as the local appearance of certain structures or colours are the most indicative features for melanoma. However despite their explanation of the importance of symmetry, they did not display a method that directly measures the symmetrical nature of a skin lesion. Instead the lesion is split into segments and these segments compared against one another, ratios are generated for the selected features perimeter, area, and form factor. Further properties of a lesion were detected by separating the lesion into a central part and a peripheral part residing on the outskirts of the lesion, colour analysis is then performed on each separated part.

While this seemed to be effective, I wished to improve upon their implementation by providing a method that directly measures the symmetrical nature of a skin lesion with a single number.

The method of measuring the symmetrical nature that I developed during this project, not only finds the optimum line symmetry of an object, but also produces a numeric output which acts as the measurement of the symmetrical nature of an object. This method was inspired by a paper written by Ho Li et al, this paper was describing a technique to perform "Real Time Detection and Segmentation of Reflectionally Symmetric Objects in Digital Images" [8]. The method of symmetry detection in this paper is similar to the method I produced, albeit with a process of rotation added.

Simply stated, the method I developed works by first finding every possible pair of edge pixels for the object, the perpendicular bisector for each pair is then considered as a potential line of symmetry. A matrix is created whose dimensions are equal the maximum polar co-ordinates possible for each given image. Polar parameterisation is then used on each potential line of symmetry to calculate the co-ordinates of the matrix which will need to be incremented. This method of voting differs from the traditional Hough Transform [11], this method requires multiple votes per edge pixel, and the voting system I have implemented however only requires a single vote for each pair of possible edge pixels. This method of voting is, like I said, is different from the traditional Hough Transform and more similar to the method used in the Randomised Hough Transform [12].

The line with the most votes is considered the optimal line of symmetry for that object, the measure of the symmetrical nature of an object is derived from the number of votes for the optimum line of symmetry, normalised by dividing the value by the number of possible edge pairs.

Algorithm Detail

The first step in this method is finding every single possible pair of edge co-ordinates possible, during the development of this algorithm I saw amounts of pairs ranging from three hundred thousand to a potential three million pair of points.

Once a list of every possible pair of points is calculated, the Cartesian equation of the line between each pair is calculated. The perpendicular bisector of these lines are the potential lines of symmetry that will be evaluated, the perpendicular bisector is a line that intersects the line at 90 degrees through the mid-point.

The perpendicular bisectors is calculated by first finding the negative reciprocal of the original line's gradient, the negative reciprocal is simply the result of swapping the numerator and denominator and changing the sign of the value (Example: $\frac{1}{3} = \frac{3}{1} = -\frac{3}{1} = -3$). With the gradient of the bisector found simple algebra is implemented using the X and Y co-ordinates of the point of intersection with the original line (its midpoint) to calculate the Cartesian line equation of the bisector.

As I have mentioned before, a matrix whose dimensions equal the maximum polar co-ordinates in the image is used to count how many times a line of symmetry has occurred. A polar parameterisation is then used to find the polar co-ordinates of the nearest point on the potential line of symmetry to the origin of the co-ordinates of the image.

Segmented Symmetry measure

As well as my previously described method of symmetry measurement, I will also be implementing similar methods to that of Ganster et al [9]. I shall use the line of symmetry calculated in the above mentioned algorithm, to separate the image into two halves and perform local colour and shape

analysis. The shape analysis methods I have decided to implement locally are, circularity, compactness, convexity, extent and solidity, these analysis methods shall be described later in detail.

Border Analysis

As I have previously mentions malignant and dysplastic skin lesions generally tend to be more misshapen and fractal in nature than that of benign lesions. But what is fractal, what does it mean to be fractal? In this section of the write up I shall discuss and explain what a fractal shape is as well as discuss a method used to measure the fractal nature of an object 'Fractal Dimension'.

What is a fractal?

The term "fractal" was coined by Benoit Mandelbrot in 1975, it is derived from the Latin word "Fractus" meaning an irregular surface such as broken bones.

A fractal is a curve or geometric figure that contains a repeating complex pattern, i.e. each part of the figure has the same statistical character as the whole

Below is an example of a universally used example of fractals, the Von Koch Curve and how it is generated from a straight line.

The Von Koch Curve is generated by a recursive process below:

1) Split each straight line into three equal segments

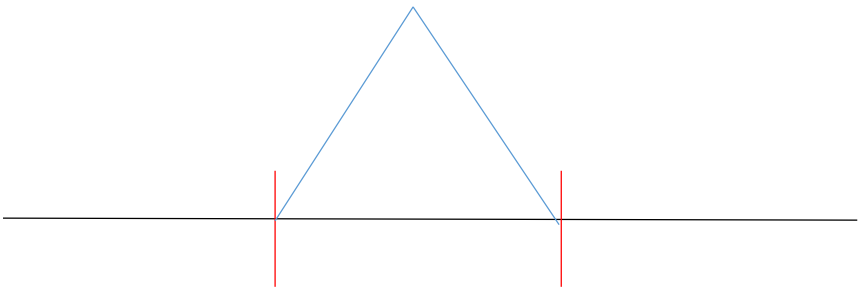
2) Replace the centre segment with an equilateral

3) Remove line in equilateral triangle corresponding to the original line

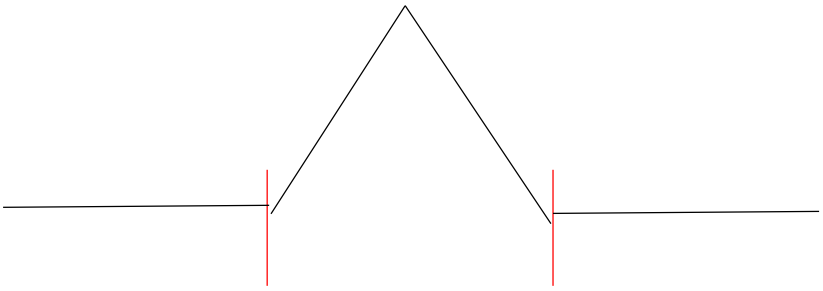
4) Repeat

← **Original Line**

Step One

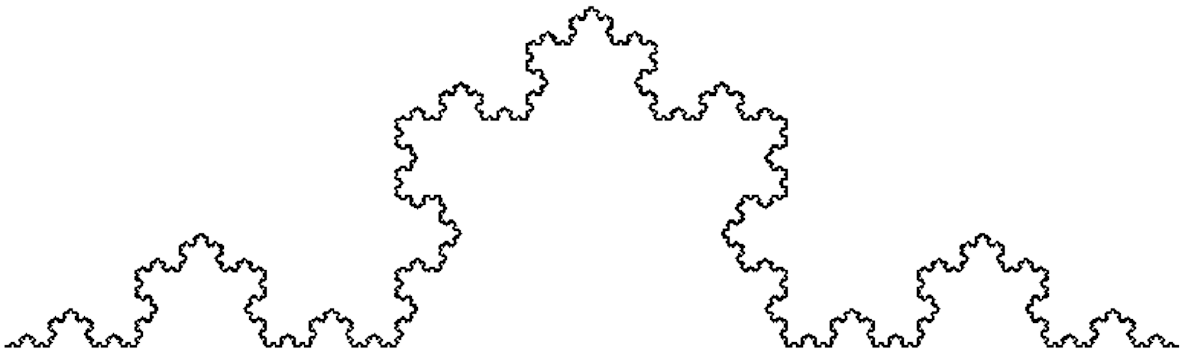


Step Two



Step Three

Repeating this recursive process would eventually result in the Von Koch Cure shown below:



Fractal Dimension

But what is fractal dimension? And how can I use it to measure an object's fractal nature? The Fractal Dimension measure is a statistical index of complexity indicating how the properties of a pattern would change when scaled.

In order to more clearly and simply explain this I shall first go over simpler shapes such as the length of a straight line and the area of a square by scaling down.

First I shall define the equation used to derive the fractal dimension, namely:

$$\left[\frac{1}{R}\right]^{D} = N \qquad D = \frac{\ln(N)}{\ln(1/R)}$$

R = Scaling Factor Ratio

N = Number of Sections

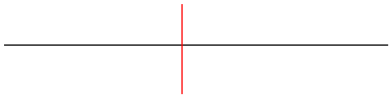
D = Fractal Dimension

Corey White – C1133127

So let's first look at a straight line

R	N
1	1
½	2
¼	4

This line has not been segmented so R and N = 1, if we were to plug them into the equation
D = ln(1)/ln(1/1) = 1

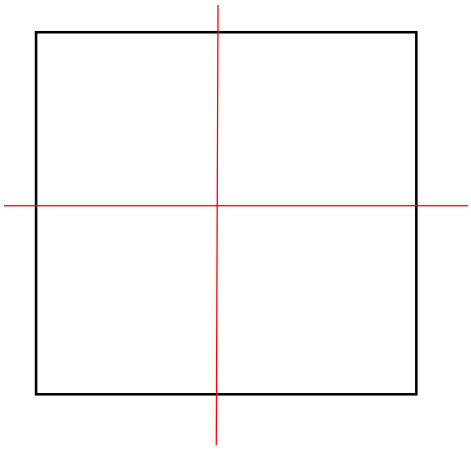


This line has been segmented into two halves, therefore R = ½ and N = 2, if we were to plug them into the equation
D = ln(2)/ln(1/½) = 1

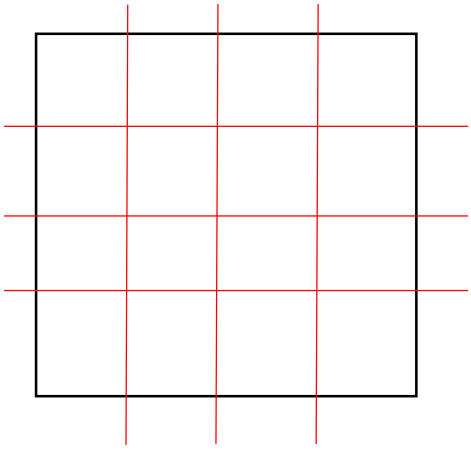
This line has been segmented into four quarters, therefore R = ¼ and N = 4, if we were to plug them into the equation
D = ln(4)/ln(1/¼) = 1

And now let's look at a square's area

R	N
½	4
¼	16



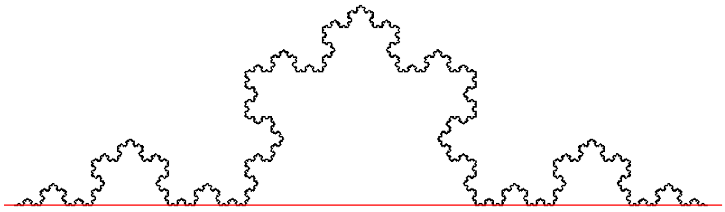
Each line in this square has been split in half therefore R = ½ and there are 4 segment squares so N = 4, if we were to plug them into the equation
D = ln(4)/ln(1/½) = 2



Each line in this square has been split in quarters therefore R = ¼ and there are 16 segment squares so N = 16, if we were to plug them into the equation
D = ln(16)/ln(1/¼) = 2

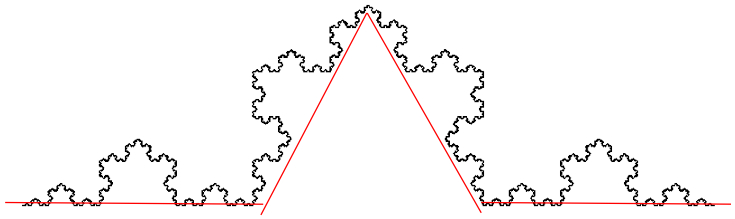
As you can see, for simple values such as the length of a straight line (or perimeter of a simple shape) has a fractal dimension of 1 and the area of simple shapes as shown above has a fractal dimension of 2. As we are going to be dealing with two – dimensional objects (when the lesions are extracted) the fractal dimension of the lesions shall fall between 1 and 2.

Below is an example of how the fractal dimension is calculated from a more complex fractal image, the Von Koch curve seen previously.

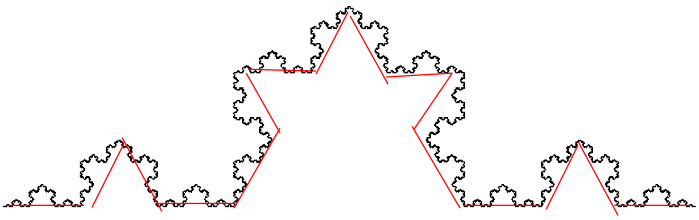


Here only a single line is used, this ignores all the detail of the complex fractal shape, R = 1 and N = 1, if we were to plug them into the equation
D = ln(1)/ln(1/1) = 1

R	N
1	1
1/3	4
1/9	16



Here the line has been segmented into 3 segments, more detail has been accommodated, R = 1/3 and N = 4, if we were to plug them into the equation
D = ln(4)/ln(1/1/3) = 1.261859……



Here each line has been segmented into a further 3 segments, even more detail has been accommodated, R = 1/9 and N = 16, if we were to plug them into the equation
D = ln(16)/ln(1/1/9) = 1.261859……

Algorithm

To calculate the fractal dimension of each skin lesion I have used the Haussdorf algorithm (otherwise known as the 'Box Counting Algorithm'). The code used for the measurement of fractal dimension was taken from MATLAB File exchange, all credit for this algorithm should go to its author.

1) Set scaling factor ratio limit

2) Set scaling factor ratio to 1

3) Divide image into segments or 'boxes' using the scaling factor ratio

4) Count the number of boxes that contains an edge pixel

5) Store scaling factor ratio and number of containing boxes into an array

6) Increment scaling factor ratio

7) If scaling factor ratio < limit return to step 3

8) Calculate dimension for each set of values in array, using equation shown below,

$$D = \frac{\ln(N)}{\ln(1/R)}$$

9) Fit a first degree polynomial

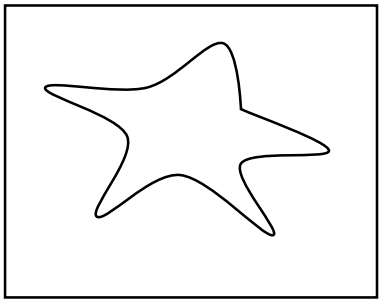
10) The first co-efficient is the object's fractal dimension

Example

Scaling factor ratio limit = 5

Scaling factor ratio = 1

Split image below:

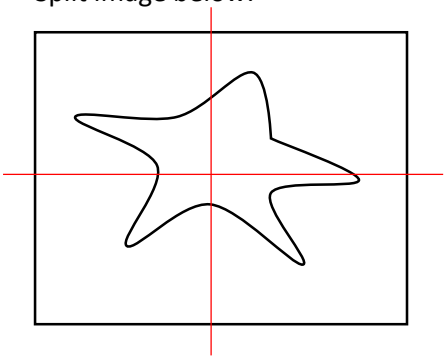


Number of containing boxes = 1

↓

Scaling factor ratio = 2

Split image below:

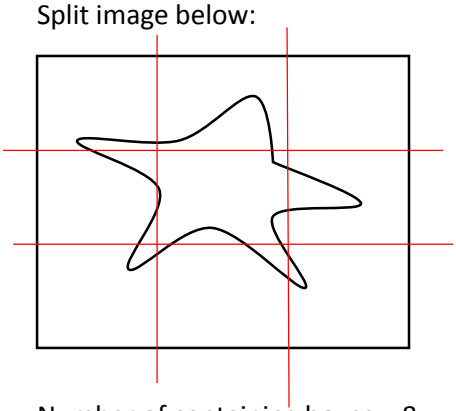


Number of containing boxes = 4

↓

Scaling factor ratio = 3

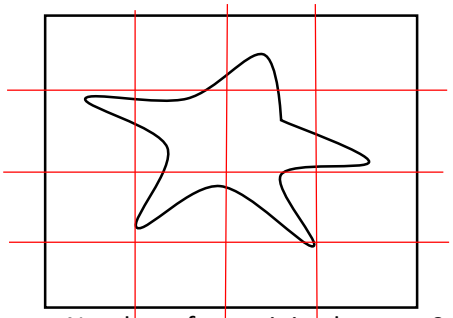
Split image below:



Number of containing boxes = 8

Scaling factor ratio = 4

Split image below:

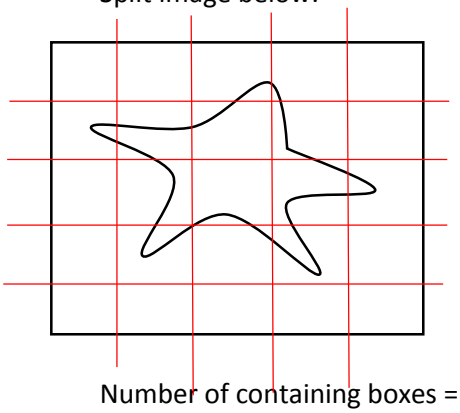


Number of containing boxes = 9

↓

Scaling factor ratio = 5

Split image below:

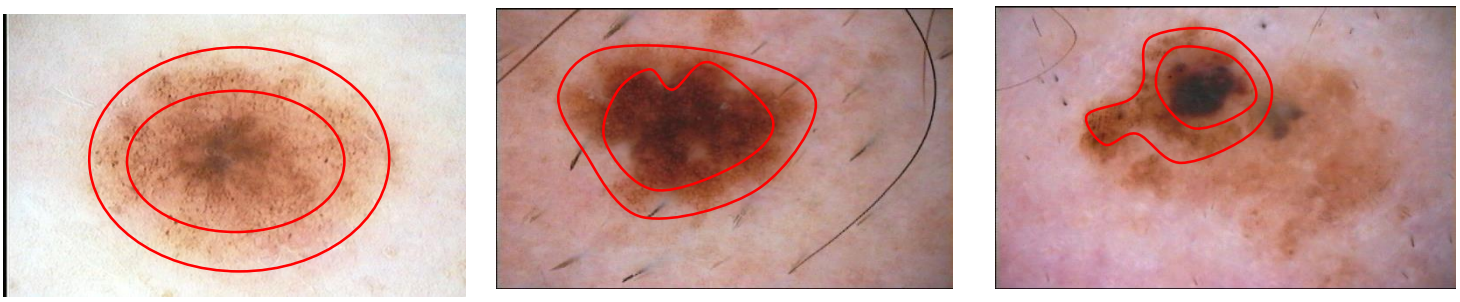


Number of containing boxes = 13

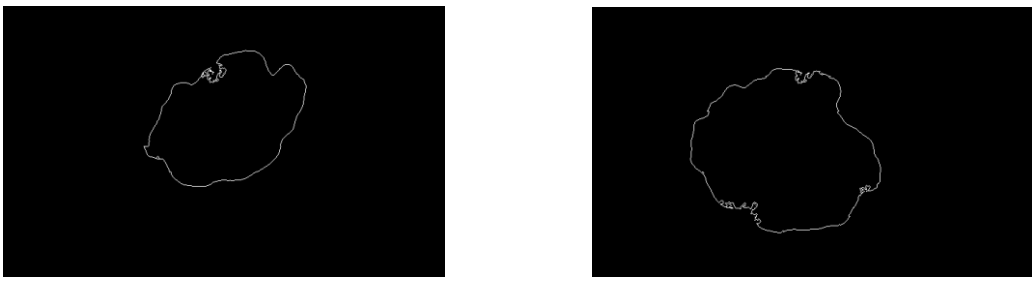
Border mask analysis

Another trait typically found at the border of malignant and dysplastic melanoma skin lesions is a more pronounced blurring of the colour and gradual colour change from the outskirts of the skin lesion toward the surrounding skin [6].

Below are a few examples:



In order to analyse the gradual change in colour, I first find the mask of the lesion using the segmentation method that I mentioned previously. Once I have obtained the binary mask, I find all the pixels that reside on the edge of that mask and change all other pixels value's in the image to zero, resulting in an image like those shown below.



I then iteratively thicken the lines shown above, both outwards to the surrounding skin area and inward toward the lesion. This provides a new mask exclusively for the analysis of the changes between the outskirts of the lesion and its surrounding skin.

With these new 'Border Masks', local analysis around the lesion's edge is now possible, by extracting the pixel values that reside within the white sections of the mask above, colour analysis of both the HSV and RGB colour space will be performed. This will provide a description of the colour transition from the lesion toward the surrounding skin area.

Compactness Index

Shape compactness is a simple and intrinsic characteristic of the shape of an object, the importance shape compactness as describing characteristic has been previously demonstrated in the field of computer vision [23].

The simplest and most widely used measure for shape compactness is given by:

$$C = \frac{P^2}{A}$$
$$C = Shape\ Compactness$$
$$P = Perimeter\ of\ Object$$
$$A = Area\ of\ Object$$

The equation above is derived from the isoperimetric inequality:

$$p^2 \geq 4\pi A$$

The isoperimetric inequality shows that for a given perimeter the shape that encloses the greatest area is a circle, therefore it holds that $p^2 = 4\pi A$ if and only if the shape given is a circle. The logical conclusion shows that a circle is the most compact shape possible. Therefore if the above equation is used to calculate the compactness of an object, the object is being compared to a circle who has the same perimeter as the object being analysed. [23]

The above equation for shape compactness can then be normalized in order for its results to fall within the range of 0 and 1 (1 being a perfect circle). Below is the normalised equation:

$$C = \frac{P^2}{4\pi A}$$

Benign skin lesions are known to typically exhibit a round shape, whereas malignant skin lesions are known to be misshapen and have irregular borders. Therefore benign and malignant skin lesions may be distinguished by their similarity to circles, using the above equation, if a shape compactness of a skin lesion is closer to 1 then it is closer to the shape compactness of a perfect circle and therefore more likely to be benign. However the closer the resulting shape compactness is to zero the less like a perfect circle the lesion is and likely to be malignant. It should also be noted however that shape compactness is sensitive to noise located at the boundary of the object, noise from lesions in this project may include body hair, an irregular skin texture (i.e. dry skin) or ulceration of the skin surface which may arise in later stages of melanoma. Another shape analysis measure I took from The Image Processing Handbook was form factor, however form factor is given by [4]

$$FF = \frac{4\pi A}{P^2}$$

As you can see, the form factor is simply the reciprocal of the shape compactness, therefore it would be unnecessary to implement both.

Colour Analysis

Global statistical Analysis

The colour of a melanoma skin lesion is another important feature, most benign skin lesions generally tend to be uniform of colour (usually brown) throughout the whole lesion. Malignant or Dysplastic lesions however typically contains a variety of colours

The feature descriptors of colour are mainly statistical in nature, parameters calculated from the different colour channels of the image. In this project I have calculated the average, variance, maximum and minimum values of the hue and intensity channels from HSV colour space of the images, because of the lighting conditions used in creating digital ELM images the saturation colour channel of the HSV colour space would provide little to no useful feature values.

I have also calculated the average, variance, maximum and minimum values from each colour channel of the RGB colour space, I would expect that the features values extracted from the HSV colour space would provide more accurate results however I will experiment with both colour spaces to see which will be more accurate.

The colour skin pigmentation of the skin lesions is dependent on the patient's skin type it would be advisable to normalize the colour channels of the image. I performed the normalization of the skin by using two methods, the first of which is taken from Ganster et al's publication [9]. This first method takes place in the HSV colour space, the intensity channel is normalized by subtracting the average intensity from the skin region of the image. The skin region of the image is extracted by inverting the mask created by the segmentation process and multiplying it by the original lesion images. The average intensity is then calculated from the resulting image, it is this value that is subtracted from each intensity value in the segmented lesion.

The hue channel is normalized by subtracting most common hue value in the skin region of the original image from the hue values in the segmented lesion. The skin region of the image is extracted the same way as previously described, the most common hue value is found, it is this value that is then subtracted from the hue values of the segmented skin lesion.

When both the hue and intensity colour channels have been normalized, the average, variant, maximum and minimum values from both the RGB colour space and the hue and intensity of the HSV colour space are calculated again and used as features.

The second normalisation method I performed was using histogram equalisation, using the RGB colour space, I separated the colour channels and performed the equalisation on each channel. I then recombined the separated channels, the average, variant, maximum and minimum values from both the RGB colour space and the hue and intensity of the HSV colour space are calculated.

Colour Quantisation

Another method of colour analysis that I have performed is the use of colour quantisation [22], colour quantisation is a method that reduces the total number of unique colours used in an image. This is usually performed with the intention of maintaining quality of an image whilst also reducing the memory required to store the image. In this case however I am using colour quantisation to reduce the lesion image down to 15 distinct colours. As malignant skin lesions display colours that are not usually present in benign lesions. To achieve this, I shall take the segmented RGB lesion image and perform colour quantisation once I have created the indexed colour image I shall convert it into HSV colour space and find the unique hues and intensities, these 30 values shall be taken as classifying features. In addition to the 15 distinct colours derived from the colour quantisation

process, I will also calculate the average and variant hue and intensity values to be used as classifying features.

Local Colour Analysis

In addition to the global features that I have extracted using the colour analysis techniques I have previously mentioned, I have also opted to perform colour analysis on each melanoma skin lesion at a local level. Using the line of symmetry that was calculated using the symmetry transform algorithm mentioned earlier and its perpendicular bisector I have separated each lesion into four quarters.

Diameter and Shape Analysis

As I have mentioned previously, there has been no information given as to whether the lesion elm images that I have been provided with have all been taken at a uniform distance from the patient. If the pictures were taken at different distances from the patient, it would render the use of size as a classifying feature impossible, as the diameter of a skin lesion is an important diagnostic feature used in physical examinations it would be undesirable to lose the use of such a feature. However since the images in question have been supplied by Ganster et als publication [9], and in that very publication they used area as an identifying feature, it can be inferred that the elm images must have been taken at a uniform distance.

Area

This feature is possibly the simplest feature that has been extracted from the skin lesion, using the ABCD mnemonic as a guideline for lesion analysis[6], lesions that are larger than 6mm in diameter are typically dysplastic or malignant in nature. Therefore the larger the area of a skin lesion, the more likely the lesion is of that nature.

Extent

The extent of an object can be defined with the equation shown below [4]

$$E = \frac{A}{BBA}$$

$$E = Extent$$
$$A = Area$$
$$BBA = Area\ of\ Bounding\ Box$$

The extent of an object denotes the proportion of the pixels residing in the bounding box that also reside in the object. Malignant or dysplastic ski lesions are typically more misshapen, whereas benign skin lesions are more regular in shape. Therefore the area for a benign skin lesion would be closer to the area of the bounding box and would be smaller than the bounding box for dysplastic or malignant.

Solidity

The equation below denotes how to calculate the solidity of an object: [4]

$$S = \frac{A}{CA}$$

$$S = Solidity$$
$$A = Area$$
$$CA = Convex\ Hull\ Area$$

The solidity of an object denotes the proportion of the pixels that are in the convex hull that are also in the object. This measure is very similar to the measure of extent, benign skin lesions wold have an area closer to the area of the convex hull and dysplastic or malignant skin lesions wold have a smaller area.

Convexity

The equation for the Convexity of an object is show below: [4]

$$Conv = \frac{CP}{C}$$

$$Conv = Convexity$$
$$CP = Convex\ Hull\ Perimeter$$
$$P = Perimeter$$

Convexity is another method to measure the fractal nature of the border, the less fractal a shape's border is, the closer its perimeter would be to the perimeter of the convex hull. I would be expected that malignant and dysplastic lesions would exhibit a less uniform shape and therefore have a perimeter larger than that of the convex hull, this would result

Hus 7 invariant moments

A measurement that was not implemented by Ganster et al in their publication [9], simply put Hus seven invariant moments are values derived from the objects that are invariant to scale and rotation.[24]

The first step in calculating the invariant moments is to calculate the raw moments, a raw moment of order (p + q) is defined by the following equation.

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p\, y^q\, f(x,y) dx\, dy$$

The area can be found from the 00th raw moment, as previously stated the area of the skin lesion can then be used as a feature. If a skin lesion has a diameter larger than 6mm then it is more likely to be either dysplastic or malignant, therefore the area of a lesion can be used to distinguish between the two types of skin lesion.

The co-ordinates of the centroid can be defined from the raw moments, the co-ordinates can be defined as below:

$$\bar{x} = \frac{m_{10}}{m_{00}} \bar{y} = \frac{m_{01}}{m_{00}}$$

The distance between the centre of the bounding box and the centroid can then be used as feature [9].

Central moments

The next stage in calculating Hu's seven invariant moments would be to calculate the central moments, central moments can be defined as a moment of a probability distribution about the variable's mean. Central moments are transitionally invariant, the central moments of order up to three are defined by the following equations: [24]

$\mu_{00} = m_{00}$

$\mu_{10} = 0$

$\mu_{01} = 0$

$\mu_{11} = m_{11} - \bar{x}m_{10}$ *or* $\mu_{11} = m_{11} - \bar{y}m_{01}$

$\mu_{20} = m_{20} - \bar{x}m_{10}$

$\mu_{02} = m_{02} - \bar{y}m_{01}$

$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}$

$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^{-2} m_{10}$

$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^{-2} m_{10}$

$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^{-2} m_{01}$

Normalization equation

In order to make the moments invariant to both translation and scale, the moments are normalized by dividing them by the corresponding 00th central moment. [24]

$$n_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(1+\frac{i+j}{2})}}$$

Hu's Seven Invariant moments

Hu's invariant moments are not only invariant to translation and scale but also invariant rotation. The normalised moments are then used in the following equations to calculate Hu's moments. [24]

$$I_1 = n_{20} + n_{02}$$

$$I_2 = (n_{20} - n_{02})^2 + 4{n_{11}}^2$$

$$I_3 = (n_{30} - 3n_{12})^2 + (3n_{21} - n_{03})^2$$

$$I_4 = (n_{30} - n_{12})^2 + (n_{21} + n_{03})^2$$

$$I_5 = (n_{30} - 3n_{12})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) + (3n_{21} - n_{03})(n_{21} + n_{03})(3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2)$$

$$I_6 = (n_{20} - n_{02})((n_{30} + n_{12})^2 - (n_{21} - n_{03})^2) + 4n_{11}(n_{30} + n_{12})(n_{21} + n_{03})$$

$$I_7 = (3n_{21} - n_{03})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) + (3n_{12} - n_{30})(n_{21} + n_{03})(3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2)$$

Circularity

Benign skin lesions typically exhibit a much more uniform round shape than dysplastic or malignant skin lesions, because of this, the circularity of the segmented masks can be calculated and used as a feature for classification.

There are many different ways to measure the circularity of an object, I have previously mentioned a one of t standard circularity measures in my description of Form Factor. I addition to this standard measurement, I've opted to also measure an objects circularity using a measure that uses invariant moments that I have mentioned and described above. [10]

The measure for circularity can be defined as:

$$C(s) = \frac{\mu_{00}(s)^2}{\mu_{20}(s) + \mu_{02}(s)}$$
$$s = binary\ image$$

The reciprocal of the above equation ($\frac{\mu_{20}(s)+\mu_{02}(s)}{\mu_{00}(s)^2}$) is well known in the field of computer vision, it one of the Hu's invariant moments, this makes the measurement of circularity invariant to rotation, translation and scale. [10]

The result of the reciprocal $\frac{\mu_{20}(s)+\mu_{02}(s)}{\mu_{00}(s)^2}$ is equal to $\frac{1}{2\pi}$ if the object is a perfect circle, therefore by multiplying the original equation by $\frac{1}{2\pi}$ you would get a measure of circularity ranging from 0 to 1.

$$C(s) = \frac{1}{2\pi}.\frac{\mu_{00}(s)^2}{\mu_{20}(s) + \mu_{02}(s)}$$

[10].

Feature Selection

Without first reducing the number of features some classification algorithms may have trouble classifying objects, it may be the case that some of the features calculated are in fact detrimental to the classification process. Therefore the main goal must be reducing the number of features to a smaller subset of features that optimise the classifier's performance.

Feature selection algorithms can be roughly split into two categories, "Filters" and "Wrappers", filter methods use the characteristics of the data such as distance or separability measures to evaluate and create subsets of features. Filter generally tend to be quick as they evaluate the data without involving the learning algorithm, however the filter methods generally select larger subsets and in some cases require the user to set an arbitrary limit on the number of features to be selected, this may be undesirable in this project as the aim is to limit the user's interaction as much as possible. [25]

Wrapper methods use the performance of the classifier to select which features to be selected, it attempts different combinations of the features, evaluating the accuracy of each combination. The final result is the combination of features that has the optimal accuracy, the advantage of wrappers over filters is a general increase of accuracy. However since the method uses the classifier repeatedly, wrappers are significantly slower than filters, and since the method is reliant on the classifier it is also bound to the bias of the classifier used. [25]

Methods Used

I have decided to implement two feature selection algorithms, the first is a simple Sequential Forward Search (SFS) algorithm and the other is a feature selection method of my own design based off the SFS algorithm.

Sequential Forward Search

The SFS algorithm is a simple greedy search algorithm, it starts with an empty set, and for each level it sequentially adds the element which increases the overall performance to the set. Simple pseudo code is shown below to describe the method.

1) Start with empty set E = { }
2) Select feature from list X1 → XN with best combined (XN + E) performance
3) If combined performance (XN + E) < current performance (E) and **go to 5**

   else: add XN to E and remove X from list and **go to 2**

New Feature Selection Method

The new Feature selection method I developed is quite similar to the SFS algorithm, it starts with an empty set, each of the feature's individual performance is evaluated using the classifier, and the features are then sorted, the best performing is moved to the set. The algorithm the moves sequentially down the list combining the set with the features, if the combination has a greater

performance than the current sets performance, the feature is added to the set. The result is a combination of features with the highest yielded performance.

Experiments with Training Data

As I have calculated and extracted several features from the skin lesions, I must devise a method to find the optimum combination of features that produce the greatest percentage of correct classifications. As there are (find out exactly) a lot more benign skin lesions in the provided data set, a classifier may achieve a percentage of 70ish percent by simply classifying all lesions as benign. Therefore the experiments I shall be conducting will find the percent of correct classifications of benign and malignant. As malignant and dysplastic skin lesions are much more serious than benign skin lesions they shall be given priority and therefore the percentage of correct malignant classifications will attribute more o the end result.

In order to find the best possible training strategy for the classifiers I have decided to use, I will be performing experiments with varying amounts of training data. I shall increase the amount of training data with no regard of the ratio of different lesion types, I would expect to see an increase in accuracy as the size of the training data increases.

Chapter 4
Implementation

Technologies used

I initially intended to use the C# programming language for this project, this was due to my past experience using C# with large scale projects and design patterns tailored for its use. However I found that through developing the project using C# my workload output was not efficient, I found that my progress and experimentation was too slow to complete this project to a decent standard in the time scale that was provided. The main issue I found was that C# did not naturally handle matrices efficiently and many of the simpler tasks took longer than it would in programming languages optimised for the use of matrices.

For that reason I decided to change my programming language of choice to MATLAB, I instantly saw a dramatic increase in efficiency of my work. As MATLAB naturally handles matrices I was able to more easily display image pixel values and manipulate matrices. As well as the noted ease of matrices manipulation that I have mentioned earlier there were additional advantages to the use of MATLAB for this project compared to C#.

The first significant advantage is MATLAB has numerous built in functions specifically for image processing (using the correct plugins), this would enable me to use these functions rather than “re-inventing the wheel” for each function I desired.

MATLAB is also very popular in the field of computer vision, because of this there is a large net of communities that discuss problems of computer vision and potential solutions, as I am still relatively new the field of computer vision I may find myself confused by some of the various image processing techniques. It is therefore very advantageous to have a large community where solutions to similar problems may have already been resolved.

MATLAB has a wide range of plugins to suit many project’s needs, there may be times where some functions are not within the plugins that I currently own. If ever this is the case, the expansive Math Works have provided the expansive MATLAB community with a place to exchange and share files called ‘File Exchange’.

As well as all the plugins and support from the MATLAB community, MATLAB also contains a simple and easy to use GUI creation tool. The tool allows a user to “drag and drop” elements into the GUI automatically generating the event call backs for each element added. Although the interfaces creating using this too are rather simplistic and not particularly advanced visually, its simple nature would be suitable for this project as it is the least important aspect and little time should be spent on it.

Potential Problems

Problem – Potential loss of work

With every project there is the possibility of a hardware failure or a similar event that would result in a significant loss progress in the project. Such issues could potentially set the project back by a large amount of time.

Solution

This problem may be mitigated by applying several different methods of storing the project and any other related documentation, never relying on a single method of storage reduces the risk of loss. I have decided to implement such protocols, routinely storing the project on several flash drives as well and online storage solutions such as drop box and google drive.

With each storage method I intend on utilising a file organisational method where different iterations of the project are stored in different folders, each with a 'ReadMe.txt' file describing the functionality to that date. This would give me an option to return to an earlier version of the project if problems are encountered.

While writing this essay I encountered a hardware failure, fortunately no work was lost due to the auto save function in Microsoft Word, to safe guard myself against any future hardware failures, I decided to increase the frequency of which Word would auto save documents to save every minute. If another hardware failure were to occur, the file is more likely to be recovered.

Problem – Memory Issues

Currently I am running the 32bit MATLAB R2013a student version on the main computer used to develop this project. Below is a table indicating the processing limits of different MATLAB versions and operating systems.

As you can see, the 32 bit version has the lowest processing limit, it has occurred in past personal projects to receive an "Out of Memory" Error. As this project may be dealing with very large matrices and vectors, there is a possibility of this occurring.

Solution

Fortunately there are various methods that may be used to resolve such errors, the first and easiest solution would be to use a 64 bit version of the software on another computer with greater RAM. While the simplest solution, this would require the use of lab computers that may not always be available for use due to tutorial schedules.

Other solutions include reducing the size of the data, compressing the data so that less fragmentation will occur, breaking down of larger matrices into several smaller matrices so less memory is used at any given time, ensuring there are no external constraints adversely affecting the memory accessible to MATLAB and the clearing of unnecessary variables.

Problem – Potential bottlenecks

As I will be evaluating thousands of images and extracting multiples features from each, there is a potential for the written processes to take a large amount of time that may slow the progress of the project. This could cause issues if the processes produce incorrect results, the time taken for that process to complete would have been wasted.

Solution

This is a problem that would be difficult to avoid, there are several techniques I can employ however that if not resolving the issue, minimises it to a manageable amount of time. Firstly processes that are known to take long periods of time can be scheduled to run during the night when the computer is not in use for other work. In cases where processes must run during the day other work such as research or documentation can be scheduled, as to not waste time simply waiting for functions to complete.

In addition to these scheduling solutions, there are several techniques that are commonly used when using MATLAB for projects to optimise the performance of your code. If the long processes took a shorter amount of time to complete, the bottle neck problem would be mitigated. Examples of MATLAB optimisation methods are listed below:

Vectorisation

MATLAB is optimised for operations involving vectors and matrices, i.e. performing an operation on a matrix or vector as a whole would be faster than using a for loop to iterate over each element. The process of converting a loop based algorithm to an algorithm that uses MATLAB's matrix and vector operations is called "Vectorisation".

There are several reasons why you may wish to vectorise your loop-based algorithms, the most advantageous of these reasons is that it would increase the performance of your algorithm. Vectorised algorithms often run much faster in MATLAB compared to their loop-based counter parts, as each algorithm would be ran over thousands of images it would be useful to create more efficient algorithms to decrease the likelihood of a workload bottleneck occurring.

Vectorised code also typically contains less lines than loop based code, therefore it stands to reason that with less code to write there will be a smaller chance of issues arising due to human error. This would increase my overall efficiency and output over a period of time, as less time would be spent correcting programming errors.

Finally vectorised code would look a lot more like the mathematical expressions found in a text book or paper. This would be desirable as it would enable your code to become more

accessible to people who have a basic understanding of mathematical expressions and have little experience in programming languages.

Pre-allocation of Arrays

There will be some cases however, where the vectorisation of loop-based code may not be an option, therefore these loop based algorithms must be optimised as best as possible. One method that can be used is the pre-allocation of arrays and matrices. If a loop incrementally increasing the size of a data structure with every loop, it would have an adverse effect on the overall performance of the algorithm.

Each time MATLAB is tasked with resizing the data structure used in the loop, it may spend additional time searching for larger contiguous blocks of memory, once a larger block is found, the array is transferred to the new location. By pre-allocating the data structure the task of searching for a sufficient block of memory shall only be performed once, thus reducing the number of tasks in the algorithm and increasing performance as a result.

This method while effective can only be used to its greatest potential if the size of the needed data structure is previously known. If the size is not known, the best alternative would be to estimate a size and pre-allocate an array to that estimation, if the data structure is filled and the algorithm is not complete then another estimation is calculated and the array resized, this method while not as effective as the pre allocation of the total data structure, would reduce the total number of resizes needed by the algorithm, and therefore increase performance.

Assigning Variables

MATLAB is a very flexible language, unlike most programming languages a variable's data type can be changed easily, for example a variable A = 123 could be changed to the variable A = "Hello World". While flexible and useful, it comes at the cost of a negative effect on the performance an algorithm as it would take additional time to process. Therefore it would be beneficial to assign different variables to specific data types required throughout the algorithm.

Short Circuit Logical Operators

Another method that can be used to boost the performance of an algorithm in MATLAB is the use of the short circuiting logical operators '&&' (and) and '||' (or). These operators don't evaluate an entire logical expression if the conditions of the expression have already failed or passed.

For example "A && B" represents a logical AND expression, if A is evaluated and shown to be false, then the second half of the expression isn't evaluated as the logical expression has already failed its requirements. "A || B" represents a logical OR expression, if A is evaluated and shown to be true, then the second half of the expression needn't be evaluated either as the logical expression has already passed its requirements.

If logical operators are used frequently in loop based code (as is likely to be the case in this project due to the number of images being processed), the use of short circuiting logical operators may reduce the number of tasks required for each algorithm substantially and therefore increase its performance.

Project Files

Segmentation

Otsu.m

This file is responsible for performing the popular Otsu thresholding method [5], it takes a single argument which is a grayscale image. The first step in this function is generating an intensity histogram of the grayscale image provided this is achieved using the MATLAB function 'imhist'. Each value of the histogram is then divided by the sum of the histogram to provide a probability distribution of the intensities in the image.

A for loop is then performed for each possible intensity value in the image, (1-256), the weight and mean of both sides is calculated for each intensity value. With these values the between class of the intensity is then calculated if the current maximum between-class is less than a between-class calculated it is replaced. The intensity level with the greatest between-class is then taken as the thresholding level, this is then applied to the provided grayscale image and returned.

DynThresh.m

This file is responsible for applying and combining the thresholding method previously mentioned, it takes a single argument, the original lesion image. For each colour channel of the provided RGB image, the image is first blurred to reduce noise using e MATLAB function 'imfilter'. The Otsu thresholding method [5] is then performed on each channel, the resulting masks are added to list of masks for processing later.

The next stage in this function dynamic thresholding, the first step in my dynamic thresholding method is determining the divisors of the width and height of the image greater than ten percent. These values are then used to split each colour channel image into vertical and horizontal strips, the Otsu thresholding method is performed on each strip, the results ae then combined into a mask and added to list of masks.

The original RGB image is then converted to several different colour spaces including CbCr. Lab and ntsc. Colour quantisation to two colours is then performed on the RGB and other colour space images. The resulting masks are then added to the list of masks to be processed later.

The file 'CheckMask.m' is then called using the list of masks as its argument, the result is a list of masks that do not grow into the edges of the image, masks with the largest object and masks with no holes in the object. Any masks that show no objects at all are also removed at this point. The final mask is then generated by the use of majority voting, a pixel I set to 1 f and only if more than half of the mask list agree. The resulting mask is then checked for any hoes n the object using connective component labelling as in 'CheckMask.m'.

If however, the final mask create has an area less than or equal one percent of the image, the segmentation s assumed to be incorrect, in that case, the mask with the largest area is then selected as the final mask.

CheckMask.m

This file is responsible for additional processing and checking of previously generated masks, the file takes a single argument in the shape of a cell array containing a list of masks. For each mask in the array connective component labelling is performed, his splits the object on the mask into groups. If a group grows into the edges of an image, the group is removed from the mask, of the remaining groups al but the largest is removed.
The next step in the function is filling any holes that might have occurred in the object, this is achieved by first inverting the mask. Connective component labelling is then performed, there should be one large group indicating the surrounding skin area and smaller groups that indicate the holes in the object.

After the above processing, if a mask doesn't contain any objects, it is removed from the list of masks, the remaining list of masks is then returned.

Shape and Size

Area.m

This file contains a function that calculates and returns the area of an object, the received argument is a logical matrix. The area of the object is then calculated by finding the number of all values that equal 1 in the matrix, the resulting value is then returned.

Perimeter

This file is responsible for calculating the perimeter of an object, the received object is a logical matrix. In order to calculate the perimeter the function first retrieves all edge pixels of the object, this is achieved by using the 'bwperim' MATLAB function, this returns a matrix where all edge pixels are set to 1 and all others 0. The perimeter is then calculated much like the area, by finding the number of all values that equal 1 in the matrix, the resulting value is then returned.

ConvexPerimeter.m

This file is responsible for calculating perimeter of the convex hull of an object, the function takes a single argument, the mask of the lesion image. The convex hull of the object is then calculated using the MATLAB function 'bwconvhull', this finds the convex hull of an object and creates a binary mask of that shape.

Once the convex hull I calculated its edges are found using the MATLAB function 'bwperim', this renders all pixels in the image except the edge pixels of the object 0. The number of pixels whose

values equal 1 is then found and considered the convex perimeter.

Extent.m

This file calculates the extent of an object, the function takes a single argument, the mask of the lesion image. The extent of an object is calculated using the MATLAB function 'regionprops', the function takes two arguments, the matrix containing the object and a string to specify which property to measure, in this case 'Extent'. The region props function returns a MATLAB structure of all the calculated Extent values, the structure is then converted into an array using the MATLAB function 'struct2array'. When the structure is converted the maximum value in the array is found and returned as the object's Extent.

Solidity.m

This file calculates the Solidity of an object, this function takes a single argument, the mask of the lesion image. The function works exactly the same as 'Extent.m', the MATLAB function 'regionprops' is used taking two arguments , the matrix containing the object and a string that specifies which property to measure, in this case 'Solidity'. MATLAB returns a structure of all calculated Solidity values, the structure is then converted into an array using the MATLAB function 'struct2array'. When the structure is converted the maximum value in the array is found and returned as the object's Solidity.

Convexity.m

This file is responsible for calculating the Convexity of an object, the function takes a single argument, the mask of the lesion image. Convexity is calculated by simply using the equation that has been mentioned, the Convex Perimeter is found by calling 'ConvexHullPerimeter.m' using the mask as its object. The perimeter is found by calling 'Perimeter.m' using the mask as its object, the convexity is then calculated by dividing the result from 'ConvexHullPerimeter.m' by the result from 'Perimeter.m'. The value is then returned as the convexity of the object.

Calculating Symmetry and Finding Line of Symmetry

EdgeCombinations.m

This file is responsible for calculating all possible combinations of edge pixels of an object, the function takes a single argument, the binary mask of a lesion image. The first step in this function is finding the co-ordinates of every edge pixel of the object, this is achieved sing the MATLAB function 'bwperim'. Tis function changes all pixels value except the edge pixels to 0, the co-ordinates for every pixel with a value greater than 0 is then found and added to an array.

The list of indices of combinations is then found, the resulting array is then reshaped and permutated into a list of all possible combinations of edge pixels. The result is given I a 2x2xN N = number of combination) matrix which is then returned. It should also be noted that this method is fully vectorised and should have a much quicker performance than loop-based code.

BatchFindBisector.m

Corey White – C1133127

This file is responsible for finding the perpendicular bisector between two points and calculating the bisector's nearest point to the origin in polar space to be used later. It should be noted that this function is fully vectorised and capable of processing multiple pairs of co-ordinates at a high performance.

The function takes a single argument, a 2x2xN matrix containing N number of co-ordinate pairs (which are calculated from the 'EdgeCombination.m file). The first step in this process is to calculate an array of all the gradients and midpoints for each pair of co-ordinates in the argument, at this point the argument given is no longer necessary and therefore is cleared from memory to preserve space.

As we are calculating the perpendicular bisector of each pair of co-ordinates, we find the negative reciprocal of all gradients in the previously calculated array, these are the gradients of the perpendicular bisector. With the midpoints and gradients of each point pair of co-ordinates simple algebra can be applied to create an array of all the Y intersects for each pair of co-ordinates.

The next step would be calculating the point on the bisector that is nearest to the origin of the image (0,0). The shortest distance between two points is a straight line therefore the shortest intersecting line starting from the origin would be perpendicular to the bisector. Hence the gradient of the line from the origin would be negative reciprocal of the bisector's which incidentally is equal to the initial array of gradients calculated. This gives us two line equations for the bisector and the line beginning from the origin, to calculate the intersection for these lines they are solved simultaneously, a simple example is shown below:

$Bisector\ Equation\colon\ y = 3x + 2$

$origin\ line\ gradient\colon\ -\left(\frac{1}{3}\right) = -\frac{1}{3}$

$Origin\ line\ Equation\colon y = -\frac{1}{3}x + 0$ ....

$$y = -\frac{1}{3}x$$

Solving the equation…

$$-\frac{1}{3}x = 3x + 2$$
$$3x + \frac{1}{3}x = -2$$
$$\frac{4}{3}x = -2$$
$$4x = -6$$
$$x = -1.5$$
$$y = -\frac{1}{3} * -1.5$$
$$= 0.5$$

It is possible however that the closest point to the origin may not be contained within the limitations of the image size therefore I established a set of rules to find the nearest point to the origin that is contained within the image.

1) All cases where X < 0, X is contained within quadrant 2 therefore add 180° to θ

2) All cases where Y < 0, Y is contained within quadrant 4 therefore add 360° to θ

3) All cases where gradient = 0, θ is set to 90°

4) All cases where θ = 0, ρ is set to its corresponding Y-intersect

5) All cases where Y-intersect is infinite, ρ is set to the X co-ordinate of its corresponding mid-point

6) All cases where Y-intersect is infinite, θ is set to 90°

The matrices of all ρ, θ, bisector gradient and bisector Y-intersect vales are then concatenated together and returned.

FindSymmetry.bmp

Simply stated this file is responsible for finding the most common row containing $\rho\ and\ \theta$ values in a matrix. It takes a single argument, the binary mask of the lesion image, this image is then used as the file 'EdgeCombinations.m' , the result is in turn used as an argument for the file 'BatchFindBisector.m' whose result is the matrix containing the $\rho\ and\ \theta$ values.

Once the most common row is found, the symmetry measure is calculated as the number of rows equal to the most common, divided by the total number of rows. The $\rho\ and\ \theta$ values are then re-converted back into Cartesian co-ordinates to be used at later, an array is the returned containing the gradient, Y-intersect, X coordinate, Y coordinate and symmetry measure.

Dissecting the mask

One of the main features that will be calculated in this projects is local shape and colour information of each lesion. To find local information on the lesion, it must first be dissected into separate sections, the following files are responsible for ultimately creating 6 different masks for different sections of the lesion.

MaskBisector.m

One of the points that the lesion is split across, is the line of symmetry calculated in the symmetry measuring method I have previously mentioned. Splitting the lesion by this line however would only allow us to split the lesion into two halves, this may not be enough for the larger lesions however. Therefore this file was created to calculate the line that perpendicularly intersects the line of symmetry through the lesion.

The first step in calculating this line, is calculating the two points on the edge of the lesion that the line of symmetry passes though. The first iteration of this process started by first finding all the edge points of the lesion using the MATLAB function 'bwperim'. The next step was creating a line mask (of the same size) where all points on the line are equal to 1, I then simply added the two masks together and the two co-ordinates that had the highest values where our two points.

A problem I encountered however was that it was possible that the line would pass through more than two points, as some of the lesions are misshapen. To correct this problem, I found all combinations of intersection points and calculated the distance between them, the two points with the maximum distance, would be the two edge points used.

In addition to there being too many intersections, there is also a possibility that only one or even no intersections will be detected, this is due to diagonal nature of pixels (example shown below).

To correct this problem, if less than two intersection points are detected, the generated line mask will be added to the original lesion mask. This would return each point of intersection though the lesion itself, the two points with the maximum distance between them is then found using the above method.

Once the two points of intersection are found, it is a simple case of calculating the perpendicular bisector between these two points, the resulting Cartesian line equation is the second equation that shall be used in separating the lesions.

SplitMask.m

This file is responsible for generating a mask from a given line equation that shall later be used to dissect the lesion's mask. The function takes a single argument, this shall be an array that includes, the gradient of the line equation, it's Y-intersect, X and Y co-ordinates (either midpoint co-ordinates if calculated from MaskBisector.m or the nearest point on the line to the origin if calculated from FindSymmetry.m) and a measure of symmetrical nature (again if calculated from FindSymmetry.m).

The first iteration of this algorithm focused on creating a split mask from more vertical lines, the maximum and minimum values along the Y-axis are used as the Y values in the equation of the line. Using simple algebra the co-ordinates of where the line crosses the edges of the image are found. These co-ordinates, along with the maximum and minimum co-ordinates of the Y-axis where X equals 0 are used in the MATLAB function 'poly2mask' to create a mask that has all 1 values on one side of the line and all 0s on the other.

This proved to be effective with the more vertical lines but would prove ineffective with lines that are more horizontally inclined. For these lines the maximum and minimum values along the X- axis are used as X values in the equation of the line. Simple algebra would again be applied here to calculate the co-ordinates of the points where the line interest the edges of the image. These co-ordinates, along with the maximum and minimum co-ordinates of the X-axis where Y equals 0 are used in the MATLAB function 'poly2mask' to create a mask that has all 1 values on one side of the line and all 0s on the other.

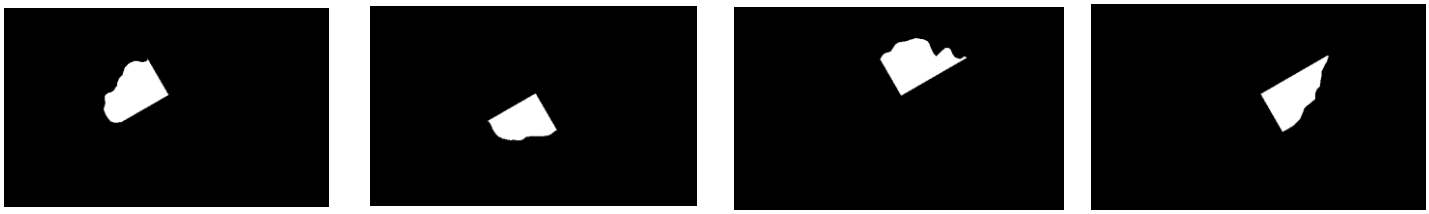
The second iteration of this function focused on categorising the two types of lines, so the correct method (both shown above) is implemented. To resolve this problem I found the absolute gradient of the line whose start and endpoints are the origin of the image and the maximum co-ordinates possible. Any line whose gradient is less than this shall be categorised as a horizontal line and any greater would be categorised as a vertical line.

An issue that arose using these methods was that bother were incompatible with perfectly vertical lines, this is because its gradient would be equal to negative infinity and its Y-intersect equal to positive infinity, and therefore the co-ordinates cannot be calculated. In these cases, the X co-ordinate value is used as each point's X value and the maximum and minimum Y values used as each point's Y value.

SeparateMask.m

This file is responsible for dissecting the skin lesions mask into two halves and four quarters, it has three given arguments, two split masks (calculated using the above processes) and the original lesion mask (created by the segmentation process). First the original mask is split into two halves by using either one of the split masks and its inverted counterpart, multiplying them with the original mask, resulting in the two masks displayed below.

Once the two half masks have been created, each are multiplied by the unused line mask and its inverted counterpart to create four quarter masks, this process is displayed below.



Border Analysis

Compactness.m

This file is responsible for calculating the Compact Index of an object, the function takes a single argument, the mask of the lesion image. The Compact Index is calculated simply by applying the equation that was mentioned previously, the perimeter is found by calling 'Perimeter.m' using the mask as the argument. The area of the lesion is then found by calling 'Area.m', using the mask as its argument. The two values are then entered into the equation

$$C = \frac{P^2}{4\pi A}$$

The result is then returned as the compact index of the object.

BorderAnalysis.m

This file is responsible for creating a mask of all the edge pixels of an object and expanding the mask both inwards towards the lesion and outward towards then skin area for later use in colour analysis. It takes two arguments, the first argument is a logical binary mask of the lesion, and the second argument is the number of iterations the user would like the algorithm to perform.

For each interaction the algorithm finds the co-ordinates of all pixel values that are greater than zero, for each of these pixels their local neighbourhood pixels are checked, if these pixels are equal to zero they are changed to 1. A problem encountered is the possibility that the border may grow into the edge of the image, any attempt to change a value beyond the scope of the image would result in an out of bounds exception, therefore additional checks were put in place to stop the algorithm attempting to access out of bounds values. The algorithm is repeated until the number of specified iterations is met, the resulting mask is then returned for later use.

Colour Analysis

ColourQuantAnalysis.m

This file is responsible for calculating the number of unique colours in a skin lesion, as well as statistical information regarding a set of quantised colours. The function takes two argument values, the original RGB lesion image and the binary mask created during segmentation. The first step is to create an array that contains all the values RGB values within the lesion itself, this is achieved by finding all co-ordinates in the mask that are equal to 1, the corresponding RGB values are then added to the array. The array is created with the following code:

```
r = im(:,:,1); g = im(:,:,2); b = im(:,:,3);
comb = cat(3,r(mask == 1),g(mask==1), b(mask==1));
```

The next step is the performance of colour quantisation on the created array using the MATLAB function 'RGB2ind'. RGB2IND takes three arguments, the first is the matrix you wish to perform

colour quantisation on, the second is the number of colours you wish to be in the colour map and the final is a string specifying whether dithering shall occur. Once the image has been converted to an indexed image, it is then converted first back to an RGB image (using the MATLAB function 'IND2RGB') and then from an RGB image to a HSV image (using RGB2HSV). It should be noted that this reconversion to RGB colour space is a necessary step as there is no function that directly converts an indexed image into a HSV image.

This process is done twice, the first quantising to 255 colours, the number of unique hues is then taken as the number of colours in the segmented skin lesion. The second time this process is used, the average, variance, minimum and maximum value is found for the Hue and Intensity channel of the HSV colour space.

HSVColourAnalysis.m

The file is responsible for calculating statistical information of the Hue and Intensity channels of the HSV colour space. The function requires two arguments, the original RGB lesion image and the lesions binary mask created by segmentation. The first step converts the original RGB image into HSV colour space, this is achieved using the MATLAB function 'RGB2HSV'.
The hue and intensity colour channels are then extracted, the values on each channel that correspond to the co-ordinate values that equal 1 in the mask are then added to an array.
The average, variance, maximum and minimum values of each array are then calculated for use as features.

RGBColourAnalysis.m

The file is responsible for calculating statistical information of the Red, Green and Blue channels of the RGB colour space. The function requires two arguments, the original RGB lesion image and the lesions binary mask created by segmentation. The first step is extracting the colour channels from the image, the values on each channel that correspond to the co-ordinate values that equal 1 in the mask are then added to an array.
The average, variance, maximum and minimum values of each array are then calculated for use as features.

NormaliseRGB.m

This file is responsible for normalising the colour of the skin lesion in RGB space, it takes two arguments, the original colour image and the binary mask created from the segmentation process. The function first separates the RGB colour spaces into three matrices, the corresponding values of where the object resides in the mask then undergoes histogram equalisation using the MATLAB function 'histeq'. The resulting matrices are then recombined to a sing image and returned.

HSVnormalise.m

This file is responsible for normalising the colour of the skin lesion in HSV space, it takes two arguments, the original colour image and the binary mask created from the segmentation process. The RGB image is first converted to a HSV image using the MATLAB function 'rgb2hsv', the hue and intensity colour channels are then separated from the image. The most probable hue from the area where the object resides in the mask is then calculated from the hue channel and mean intensity from the area where the object resides in the mask is then calculated from the intensity channel.

The most probable hue and mean intensity s then subtracted from their appropriate channels where the object resides on the mask. The HSV channel are then re-combined to a HSV image, and re-converted to n RGB image using the MATLAB function 'hsv2rgb'.

Hu's Invariant Moments

Rawmoments.m

This file is responsible for generating the raw moments for an object up to the 3rd order moments, it takes a single argument, the logical binary mask of a skin lesion. The first step in this function is creating a mesh grid from 1 to the width and height of the object, this is achieved using the MATLAB function 'meshgrid (1:width, 1:height)'.

The first moment is a calculation of the area of an object, if the area is equal to zero, the first moment is set to epsilon.

The other raw moments are then calculated using the moment equation that has been previously mentioned, the results are added to a scalar MATLAB structure with variable names similar to the moments equation, the structure is then returned for later use.

CentralMoments.m

This file is responsible for calculating the central moments of an object, central moments are invariant to translation, the function shall calculate all central moments of order up to three. The function takes a single arguments, the logical binary mask of a lesion, this function requires the raw moments of an object, and therefore a structure of them is created using the 'RawMoments.m' file.

With the structure of raw moments, the $\bar{x}\ and\ \bar{y}$ is calculated using the 0 and 1st order moments (it should be noted that $\bar{x}\ and\ \bar{y}$ are the co-ordinates of the object's centroid. The 0th central moment is the same as the 0th raw moment, the 01 and 10 central moments are always zero and are therefore not calculated. The rest of the central moments are then calculated using the equations that have been mentioned earlier, the results are given in a scalar MATLAB structure and returned.

Normalize.m

The purpose of this file is to normalise the previously calculated central moments in order to make them invariant t both translation and scale. The function takes a single argument, a scalar MATLAB structure, each element of the structure is then normalized by the appropriate figure given from the normalisation equation shown earlier. As the same order of moments are being calculated throughout the project, the normalisation divisor may be static. The 11, 02, 20 order central moments are each divided by 2 and the 30, 03, 21, 12order central moments are divided by 2.5, the normalised structure is then returned for later use.

HusMoments.m

This file is responsible for taking the normalised central moments previously found and calculating the famous Hu's seven invariant moments, this would make the moments not only invariant to scale and translation, but also invariant to rotation. This is achieved by simply applying the normalised central moments correctly in the shown equations. Some of these equations were particularly long and therefore were prime for human error, I therefore calculated the moments by paper and compared the results for assurance.

Circularity.m

This file is responsible for calculating the circularity of an object, using Hu's invariant moments, it takes a single argument, the logical binary mask of the lesion image. The first step in calculating the circularity is calculating the non-normalised central moments of the object, this is achieved by calling the file 'CentralMoments.m'. The non-normalised central moments are then used in the equation shown earlier, this results in a measure of circularity which is then returned. [10]

Chapter 5
Results and Evaluation

Segmentation

The results of the segmentation method I implemented were generally positive, of a total 4500 images approximately 302 images were incorrectly segmented, this resulted in a 93.21% correct segmentation rate. The incorrect segmented images were then excluded from the training and testing process.

I was initially disappointed with the results of my segmentation method as it showed less accurate results than that of Ganster et al's paper [9], which displayed results of 96%. However my segmentation did achieve greater results than that of [19] which displayed results of 82.4%%, showing more than 10% difference in accuracy.

The segmentation method I produced struggled with two types of images, the first type is skin lesions that were either very close or actually growing into the edge of the image. One of the rules I implemented while evaluating the generated masks in fact discouraged this type of image.

Classification Experiments

Sequential Forward Search

The following is the results from the sequential forward search of the features, I shall be showing the features that were selected as well as the performance for malignant, benign and overall.

1000 Training Images, 4500 Test Images

SFS Selected Features
Quarter RGB Analysis
Quarter Circularity
RGB Normalised HSV Analysis
RGB Colour Analysis
RGB Normalised RGB Analysis
Half RGB Analysis
Border RGB Analysis
Quarter Solidity
Quarter Extent
RGB Normalised Quantised Analysis
Hu's First Invariant Moment
Quarter Compactness
HSV Normalised RGB Analysis
Half Extent
RGB Normalised Quantised Number
Quarter Convexity
Hu's Sixth Invariant Moment

Malignant Classification Percentage: 67.65%

Benign Classification Percentage: 65.15%

Average Classification Percentage: 66.4%

2000 Training Images, 3500 Test Images

SFS Selected Features
RGB Colour Analysis
Quarter RGB Analysis
Solidity
RGB Normalised RGB Analysis
RGB Normalised Quantised Analysis

Malignant Classification Percentage: 66.78%

Benign Classification Percentage: 64.54%

Average Classification Percentage: 65.66%

3000 Training Images, 2500 Test Images

SFS Selected Features
RGB Colour Analysis
RGB Normalised Quantised Analysis
RGB Normalised HSV Analysis

Malignant Classification Percentage: 70.16%

Benign Classification Percentage: 62.89%

Average Classification Percentage: 66.51%

4000 Training Images, 1500 Test Images

SFS Selected Features
RGB Colour Analysis
Symmetry Measure
Solidity
RGN Normalised RGB Analysis
RGB Normalised Quantised Analysis
RGB Normalised Quantised Number

Malignant Classification Percentage: 75.51%

Benign Classification Percentage: 51.09%

Average Classification Percentage: 63.3%

5000 Training Images, 500 Test Images

SFS Selected Features
RGB Colour Analysis
RGB Normalised RGB Analysis
RGB Normalised Quantised Analysis
RGB Normalised HSV Analysis
Quarter RGB Analysis
Quarter Circularity
Quantised Number

Malignant Classification Percentage: 78.09%

Benign Classification Percentage: 62.96%

Average Classification Percentage: 70.53%

Custom Feature Selection

The following is the results from the custom feature selection algorithm, I shall be showing the features that were selected as well as the performance for malignant, benign and overall.

1000 Training Images, 4500 Test Images

Custom Selected Features
Perimeter
Quantised Colour Analysis
HSV Colour Analysis
Solidity
Extent
Half Complexity
Half Solidity
Hu's Forth Invariant Moments

Malignant Classification Percentage: 65.15%

Benign Classification Percentage: 67.9%

Average Classification Percentage: 66.53%

2000 Training Images, 3500 Test Images

Custom Selected Features
Area
Extent
Half Compactness
Solidity
Convexity
Half RGB Analysis
Perimeter
Half Extent
Half Circularity
Quantised Number
Compactness
Hu's Seventh Invariant Moment

Malignant Classification Percentage: 64.54%

Benign Classification Percentage: 66.93%

Average Classification Percentage: 65.74%

3000 Training Images, 2500 Test Images

Custom Selected Features
Quarter HSV Analysis
Area
Convexity
Symmetry Measure
Half Circularity
Half Extent
Half Solidity
Quarter Extent
Quarter Convexity
HSV Normalised Quantised Number
Hu's Seventh Invariant Moment
Half Compactness

Malignant Classification Percentage: 66.88%

Benign Classification Percentage: 70.06%

Average Classification Percentage: 68.47%

4000 Training Images, 1500 Test Images

Custom Selected Features
HSV Colour Analysis
Convexity
HSV Normalised Quantised Number

Malignant Classification Percentage: 51.09%

Benign Classification Percentage: 75.76%

Average Classification Percentage: 63.43%

5000 Training Images, 500 Test Images

Custom Selected Features
Quarter HSV Analysis
Circularity
Perimeter
Half Convexity
Quarter Extent

Malignant Classification Percentage: 72.84%

Benign Classification Percentage: 68.81%

Average Classification Percentage: 70.83%

Evaluation

From the above analysis you can clearly see that features revolving around the colour analysis and the shape analysis of a lesion generally tend to be the features with the height level of performance.
I was disappointed that the fractal dimension of a skin lesion didn't produce a higher level of performance, however I think the reason for fractal dimension being ineffective as a feature is likely due to the pre-processing blurring. This process smoothed the edges of the lesion reducing each lesion's fractal nature and thus reducing the usefulness of the feature.
Border colour analysis also proved to be ineffective as a feature, in fact individually both Border colour analysis would classify all lesions as benign. This may be that the border mask generated with 'BorderAnalysis.m' did not produce a large enough mask to perform sufficient colour analysis between the skin area and lesion area.

An evaluation of both of the feature selection algorithms I think proves that the Sequential Forward Search in general is the better of the two feature selection features. This is because the custom feature selection while producing decent results varies too dramatically compared to the results of the SFS algorithm which is shown here to have consistent values throughout each iteration.

Chapter 6

Summary and Conclusions

The first goal of this project was to provide an effective means of image segmentation specifically tailored for the ELM skin lesion images provided. The final results for the segmentation process I developed yielded an impressive percentage of approximately 93%, while this may be slightly less than the results reported by Ganster et al (yielding a percentage upwards of 96%) [9], I would still very much consider this portion of the project a complete success.

I would attribute the success of my segmentation method to the fusion process, here I took results from multiple different segmentation algorithms, as with most walks of life it is important to draw knowledge from more than one source. If one method is unable to successfully segment a lesion then another may be more successful. Another great feature of the fusion process is the ability to further extend it by utilising additional segmentation methods (such as region growing and edge detection) in future attempts.

The main goal behind the project was to train a classifier to distinguish between the different types of skin lesion. As I chose to use the Support Vector Machine classifier which only classifies between two groups of data, I grouped together the different types of skin lesion to form two groups. Because of their increased severity and similar characteristics I grouped Dysplastic and Malignant skin lesions together, I also grouped Benign and Non-Melanocytic skin lesions into another group.

As an initial starting point I used the popular ABCD mnemonic to design my feature extraction methods. Most of the features based on this particular rule of thumb focused on the size and shape of the skin lesion and the distribution of colours in the lesion, in fact as shown in the results section when taken individually features based on the shape and colour distribution are typically the highest performing features.

It is important to reduce the number of features used in the classification process down to the combination with the greatest performance. There are many more Benign skin lesions compared to the number of Dysplastic and Malignant skin lesions in the dataset. Therefore overall correctness should not be used as to measure performance as simply categorising all lesions as benign would result in a correctness of approximately 70%. Instead I decided to use the average correctness of both Benign and Malignant skin lesions.

The final results of the project display a maximum overall percentage of 70.53%, this was rather disappointing as I would have liked to achieve similar results to that of Ganster et al [9]. Although the results were not as I had hoped, I still consider this project a success, as stated in the introduction of this paper, it is important to detect malignant melanoma in its earliest possible stage to increase a patient's possible survival rate or even cure. Therefore I shall be evaluating this projects successfulness on its ability to successfully categorise

Dysplastic and Malignant skin lesions. Although not as impressive, this project has still successfully categorised a majority of the Dysplastic and Malignant skin lesions in the test data, which as a first attempt at such a project I am particularly pleased with. The results of the classification process also remain typically homogenous throughout, showing that the positive results were not simply the result of a 'lucky guess'.

I feel the weakest part of this project is the classification experiments, I had initially planned to perform a multitude of experiments with the classification process. The classification process in this project could be improved upon by utilising the leave one out cross validation method. This method works by using a single image as the test data and the rest of the images as training data, this process is repeated until all individual images has been used as test data. This would provide a much more thorough examination of the of the classification process.

Another way the classification experiments could have been improved it by specifically selecting the images in the training data to be have a more equal distribution of different types of lesions in the data set (Non-Melanocytic, Benign, Dysplastic and Malignant). This would reduce the likelihood of a bias forming within the classifier, although it should be noted there are very few purely malignant skin lesions in the data set and therefore having an equal number of lesion types would result in a small amount of training data.

In conclusion overall I am pleased with the outcome of this project, although given foresight there are a number of aspects I would change and would also manage my time more efficiently. However I will say that this project has been a fantastic learning experience and has allowed me to further develop my skill set in my favourite topic to date.

Chapter 7
Reflection on Learning

Upon reflection, I realise that many of the aspects of this project took much longer than I initially expected them to. I partly attribute this to my lack of experience in the field, my inexperience led me to spend significantly longer implementing and understanding aspects of computer vision that would come easier to one more experienced.

A significant hindrance to my efficiency throughout the project was the compelling need to fully understand all methods that I would be implementing. As MATLAB has a large category of image processing functions, it would have been far easier to use them without first fully understanding them. Although this adversely effected my overall efficiency I would say I made the correct choice, as I now have a much greater understanding of computer vision itself and will be able to create projects such as these far more efficiently in future.

I would say this is the largest project I have ever attempted as an individual and as a result has definitely reinforced the ideal of teamwork when creating large scale projects. When I first arrived at university, I took the general practice of group work with a pinch of salt and generally thought negatively at the prospect of such tasks. However if my time at Cardiff University as taught me one thing, it is that two in harmony will always surpass one in perfection. Comparing the experience of creating this project to previous team experiences both in University and a professional working environment, the team projects were much more pleasant, less stressful and overall superior results.

A shortcoming I feel is likely typical of most students that I fell into, was my initial complacency at the beginning of the project. Toward the beginning of the project I felt as though I had all the time in the world and as a result did not effectively manage my time. Due to this complacency I created a much more stressful working environment for myself in the latter stages of the project. This had a negative effect on the classification section of my project, I initially had a multitude of classification experiments I wished to engage in, however due to my time management or lack thereof at the start of the project I became pushed for time and was therefore unfortunately unable to produce them.

Although this project has been my most difficult and at times most stressful, I would definitely say that I have enjoyed the experience overall. Computer Vision was definitely one of my favourite subjects throughout my university career and I thoroughly enjoyed taking the reins on such a large computer vision project, delving more in depth into the topic.

Future Work

Although I have achieved generally impressive results with my segmentation process in the project I would still attempt to improve upon these results were I to attempt this project again. I would argue that the segmentation process is one of the most important aspects of the whole project, in fact most of the features calculated in this paper were reliant on the objects derived from segmentation. If this process is handled incorrectly, it would cause a negative ripple effect throughout the project as many of the features would be incorrectly measured. This project heavily relied on thresholding as the main method of segmentation, given the adaptable nature of the fusion process described I would like to use additional segmentation algorithms.

One feature I was disappointed that I was unable to implement was the evolution of a skin lesion over time. The reason for this not being implemented is that I lacked images showing the same lesion over a period of time. This was particularly disappointing as I believe it would have significantly improved the classification accuracy of the project. If I were to create a similar project in future, I would request data of the same patients over a period of time, this would allow me to add evolution to the classification process which up to now has been absent in such projects. Although I do understand the likelihood of retrieving this data set would be low as it would require the consent of numerous patients to have their diagnostic information recorded over long periods of time.

In future classification projects I would not only like to employ the improvements that I mentioned in my conclusion, but also the use and experimentation with multiple classifiers. In previous attempts at lesion classification projects papers generally only used a single classifier, I would aim to improve upon that research by analysing and displaying the results of lesion classification with multiple classifiers such as the K nearest neighbour and the artificial neural network classifier.

References

1) Cancer Research UK. Skin Cancer: Key Stats. Http://publications.cancerresearchuk.org/cancerstats/statsskin/keyfactsskin.html (accessed 10 April 2015).

2) Cancer.org. What are the survival rates for melanoma skin cancer, by stage?. http://www.cancer.org/cancer/skincancer-melanoma/detailedguide/melanoma-skin-cancer-survival-rates (accessed 10 April 2015).

3) B. Chanda and D. Dutta Majumder, Digital Image Processing and Analysis. New Delhi: Prentice-Hall of India, 2000.

4) Russ JC. The Image Processing Handbook, 6th ed. : CRC Press; 2011.

5) N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Biomed. Eng., vol. BME-9, pp. 63–66, 1979.

6) Friedman RJ, Rigel DS, Kopf W. Early detection of malignant melanoma: the role of physician examination and self-examination of the skin. CA Cancer J Clin 1985;

7) H Kittler, M Seltenheim, M Dawid, et al. Morphologic changes of pigmented skin lesions: a useful extension of the ABCD rule for dermatoscopy J Am Acad Dermatol, 40 (1999)

8) W.H. Li and L. Kleeman, "Real Time Object Tracking Using Reflectional Symmetry and Motion," Proc. IEEE Int'l Conf. Intelligent Robots and Systems, 2006.

9) H. Ganster, A. Pinz, R. Rohrer, E. Wildling, M. Binder, H. Kittler Automated melanoma recognition IEEE Trans Med Imaging, 20 (3) (2001)

10) J. Zunic, K. Hirota, P.L. Rosin, "A Hu moment invariant as a shape circularity measure", Pattern Recognition, vol. 43, no. 1, pp. 47-57, 2010.

11) R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Commun. ACM, vol. 15, no. 1, pp. 11–15, 1972.

12) L. Xu and E. Oja, "Randomized hough transform (rht): basic mechanisms, algorithms, and computational complexities," CVGIP: Image Underst., vol. 57, no. 2, pp. 131–154, 1993.

13) Ritter GX, Wilson JN. Handbook of Computer Vision Algorithms in Image Algebra. Boca Raton: CRC Press, 1996.

14) Elbaum M, Kopf AW, Rabinovitz HS, Langley RG, Kamino H, Mihm Jr MC, et al. Automatic differentiation of melanoma from melanocytic nevi with multispectral digital dermoscopy: A feasibility study. J Am Acad Dermatol 2001;

15) Hintz-Madsen M, Hansen LK, Larsen J, Drzewiecki KT. A probabilistic neural network framework for detection of malignant melanoma. In: Naguib RNG, Sherbet GV, editors.

Artificial neural networks in cancer diagnosis, prognosis, and patient management. Boca Raton: CRC Press; 2001.

16) Rubegni P, Burroni M, Cevenini G, Perotti R, Dell'Eva G, Barbini P, et al. Digital dermoscopy. Analysis and artificial neural network for the differentiation of clinically atypical pigmented skin lesions: A retrospective study. J Invest Dermatol 2002;

17) Kahofer P, Hofmann-Wellenhof R, Smolle J. Tissue counter analysis of dermatoscopic images of melanocytic skin tumors: Preliminary findings. Melanoma Res 2002;

18) Iyatomi H, Oka H, Saito M, et al. Quantitative assessment of tumour extraction from dermoscopy images and evaluation of computer-based extraction methods for an automatic melanoma diagnostic system. Melanoma Res 2006;

19) Blum A, Luedtke H, Ellwanger U, Schwabe R, Rassner G, Garbe C. Digital image analysis for diagnosis of cutaneous melanoma. development of a highly effective computer algorithm based on analysis of 837 melanocytic lesions. Br J Dermatol 2004;

20) Menzies SW, Bischof L, Talbot H, Gutenev A, Avramidis M, Wong L. The performance of SolarScan: An automated dermoscopy image analysis instrument for the diagnosis of primary melanoma. Arch Dermatol 2005;

21) J. Kittler and J. IUingworth, Minimum error thresholding, Pattern Recognition 19, 41-47 (1986).

22) X Zhang, ZM Song, YL Wang, H Wang. Color Quantization of Digital Images. Proceedings of PCM2005,

23) Montero, R. S., & Bribiesca, E. (2009). State of the art of compactness and circularity measures. International Mathematical Forum, 4(25-28), 1305–1335

24) M.K. Hu, "Pattern Recognition by Moment Invariants", Proc. IRE Trans. Information Theory, vol.
pp. 179-187, 1962.

25) John G, Kohavi R. "Wrappers for feature subset selection", Artificial Intelligence, Vol.97, No.1-2, pp.272-324, 1997.

26) MathWorks. Techniques for Improving Performance. http://uk.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html (accessed 20 April 2015).

27) MathWorks. Vectorization. http://uk.mathworks.com/help/matlab/matlab_prog/vectorization.html (accessed 20 April 2015).