Smartphone App Concept for Medical Applications.

Alexander Gustafson
University of Applied Sciences,
Zürich,
Switzerland,
alex_gustafson@yahoo.de

October 4, 2016

Dozent: Reto Knaack (knaa@zhaw.ch)

School of Engineering, Abteilung Zürich
Studiengang Informatik

Abstract

The market for smartphone based medical applications is a relatively new and growing quickly. The majority of medical apps are relatively simple health management and tracking applications that might remind a user to take his or her medicine or monitor blood pressure and heart rate data provided by accompanying devices. However, more sophisticated apps can directly provide diagnostic information by capturing and analysing data directly. Several apps exist that can asses the risk of skin cancer by tracking changes in the growth of skin lesions over time.

Contents

Chapter 1

Introduction

1.1 Background

Modern smart phones have opened new possibilities in all kinds of fields. For medical applications the combination of high resolution cameras, sensor devices, powerful CPUs and mobile networking capabilities could even be life saving. By combining input devices with powerful algorithms it should be possible to provide important diagnostic information that can inform it's user of impeding risks. One particularly interesting medical application would be the detection of melanoma in a skin lesion using a mobile phone.

Melanoma is a type malignant skin tumour that develops from benign melanocytic nevi. Although less frequent than other skin cancer types, it causes more deaths [4], and it's incidence is increasing dramatically, especially in the young white population [17]. Early diagnosis and treatment is vital.

Several non-invasive techniques exist which dermatologist can employ to visually make a diagnosis. The most common are pattern analysis, the ABCD rule, the 7-point checklist and the Menzies method, among others [13]. These methods generally use a combination of defined visual clues and indicators to assess the risk of a skin lesion. CAD (Computer Aided Diagnosis) systems exists that can assist a dermatologist, and even provide automatic diagnosis with an accuracy comparable to that of doctors trained in dermoscopic techniques [9].

1.2 Project Description

This bachelor project will investigate the following questions:

- What smartphone based medical apps are available that can track and assess the risk of skin lesions?
- What methods and algorithms can be employed by mobile phones to calculate the risk?

- What is necessary to build a mobile app that can provide a risk assessment of skin cancer melanoma from captured images?

1.3 Goals

1. Research the medical app market to gain an overview of areas of development, what apps are available, what trends are discernible. Of special interest are apps that use internal sensors (e.g., camera) and mathematical algorithms to detect skin cancer melanoma.
2. Gain familiarity with image processing and analysis algorithms. Describe the basic principles of these algorithms.
3. Research and compare available data and algorithms (e.g. in computer vision / machine learning). Define the relevant attributes that are evaluated for the analysis. This can also include changes in size over time. Optionally use machine learning techniques to try to improve the results of the analysis algorithms
4. Based on the research and comparison choose a specific algorithm and data with which to proceed. The algorithm must be able to
   (a) identify the skin lesion correctly,
   (b) compute the relevant attributes,
   (c) use these attributes to classify the skin lesions whether it is possibly cancerous or not
5. Implement the algorithm as a prototype (using Python for example). Using images of a skin lesion as input, the algorithm will provide a risk assessment as output. Implement tests that can calculate the accuracy of the algorithm. Compare the implemented algorithm with at least one other algorithm.
6. Create a concept for a medical mobile app that can utilise the features of modern smartphones to provide a risk assessment of skin cancer melanoma based on captured images. This includes a full requirements analysis, i.e. use cases, functional / non-functional requirements, architecture.
7. Implement as proof of concept specific aspects of the medical mobile app.

1.4 Expected Results

1. Results of the medical app market research including graphical presentation of trends and statistics.
2. Documentation of the available data and algorithms including high level excursion into the theory behind image processing and analysis algorithms.

3. Comparison of available data and algorithms. Definition of relevant attributes.
4. Decision, which algorithm to use.
5. Documentation of the implementation of the algorithm, including tests, evaluation and comparison with at least one other algorithm.
6. Documentation of a concept for a medial app, including defined use cases, requirements, architecture and implemented design patterns.
7. Proof of Concept: implementation of specific aspects of the medical mobile app.

Chapter 2

Market Research

2.1 Data Gathering

2.1.1 Gathering Data from the Apple iTunes Store

Searching the Apple iTunes store is typically done manually via the iTunes Application from which text and data cannot be automatically extracted. Therefore, searching for and gathering data about IOS Applications is not easy. However, Apple does provide an rss feed that can be used to list Apps in specific categories and ordered according to how new, or how popular they are and if they are free or not. The rss feed is limited to 100 items per category. The data provided by the rss feed is minimal, not much more than title and a text description of the app. There are no sub-genres or tags than can be used to further differentiate the apps.

Using a python script data was gathered from the following rss feeds:

Top 100 Free Medical Apps Top 100 Grossing Medical Apps Top 100 Paid Medical Apps This combined results included data about 255 IOS apps. The title and description fields were imported into a database. Other information from the data such as price, right, or image link were ignored.

2.1.2 Gathering App Data from the Google Play Store

In order to gather data from the Goole app store a script was programmed that could extract lists of apps from a specific url. The following urls were scanned:

- Top Paid Medical Apps : https://play.google.com/store/apps/category/MEDICAL/collection/topselling_paid
- Top Free Medical Apps : https://play.google.com/store/apps/category/MEDICAL/collection/topselling_free

For each app listed the script would extract the url of the app's detail page. From the detail page more imformation would be gathered and stored in a database. The data set is similar to that of the itunes rss feed. The title and description text were imported, other fields such as pricing and copyright were ignored.

Data on 480 Medical Apps for Android was imported. However a significant percentage of the apps could not be classified because the description text was in a language other than English, German, or French.

2.2 Categorization

The term ”Medical App” is broad and neither the iTunes nor Google app stores offer any kind of sub categorization. In order to get a better overview of what sort of Medical Apps are available it was necessary to manually browse the gathered data and assign categories to the apps.

A database management tool was created using the python based Django Web Framework. Django provides many tools that makes constructing and interacting with databases very easy. The built-in backend administration tool can be configured to browse, edit, and filter data.

In order to quickly browse through and categorize over 700 apps. The Django backend admin was configured so apps could be categorized one after the other with a minimum of clicks or scrolling. The user was presented a list of uncategorized apps. The first one is clicked. The user is then presented with a page displaying the title and summary text of the app and a field from which a category can be selected. Once saved, the app is no longer presented on the list, the user can select the next app at the top of the list.

2.2.1 Description of Categories

- **Community** - provides some soft of social networking service through which the user can share data with her family or with a network of people suffering from similar disorders.
- **Fun / Entertainment** - These apps have to real medical purpose. They are for enjoyment only.
- **Alert / First Response** - Apps that assist first responders or that help users alert first responders that help is needed.
- **Health / Lifestyle** - Relaxation and meditation apps, or ovulation and fertility reminders
- **Resource Finder** - Apps that locate resources in the vicinitly, nearby pharmacies or care providers.
- **Reminder** - Apps with timer or calendar functionality that might remind a user of an appointment or manage medince consumption.
- **Algorithmic / Diagnostic** - These are apps that provide some sort of diagnositc information based on data that as been gathered by sensors or entered by the user. Examples are seizure detection apps, or stroke severity evalutaion apps.

- **Learning / Educational / Reference** - By far the largest category, this includes apps that provide reference information about diseases or education material like anatomy apps for example.
- **Organisational** - Apps in this category might help a user or practitioner organise, share or track data and documents. Examples are apps that help users track the status of their blood pressure or blood sugar levels, create health diaries, or manage clinical data and images.

2.3 Results

2.3.1 IOS Medical Apps

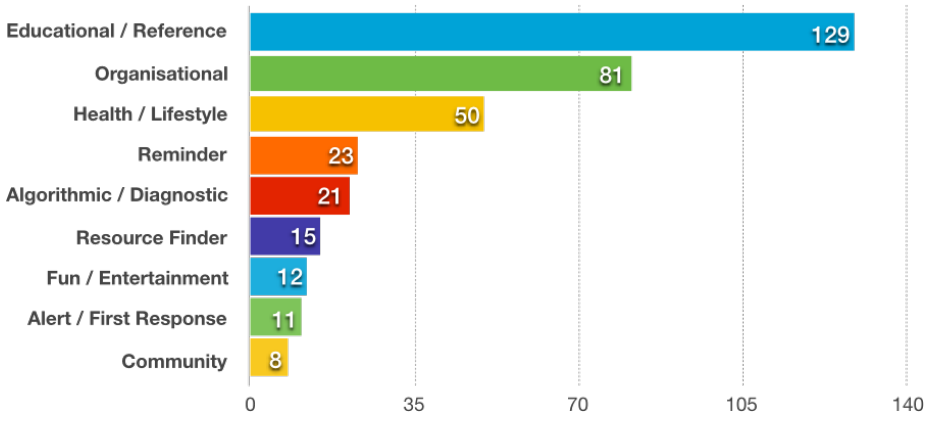

Figure 2.1: Medical Apps on the iTunes Apple Store, Search conducted on 17.05.2016

2.3.2 Android Medical Apps

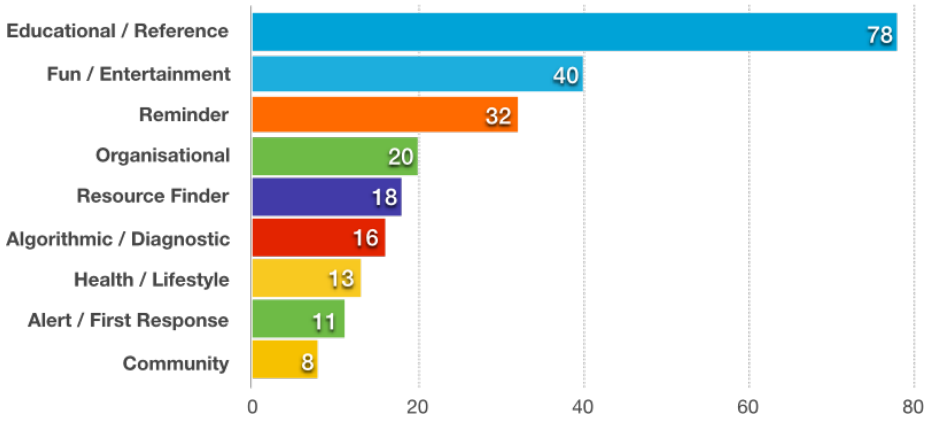

Figure 2.2: Medical Apps on the Goolge Play Store, Search conducted on 17.05.2016

2.3.3 Dermatology Apps 2013 vs 2016

Mobile apps are an especially good fit for dermatology-related care. Most dermatological conditions are by nature visible. The initial diagnosis and follow up monitoring is mostly done visually. A mobile device with a camera can aid patients and practitioners in the diagnosis of a dermatological condition and tracking it's development. The article Mobile Applications in Dermatology [8] in 2013 identified 229 dermatology-related apps across 5 app platforms (Android, Apple, Blackberry, Nokia, and Windows). These were grouped into categories based on their primary functionality. The "Self-surveillance/diagnosis" category was the second largest on the Android and Apple platforms, with 13 and 24 apps respectively.

Table 1. Total Applications

Category	Android	Apple	Blackberry	Nokia	Windows	Total, No. (%)
Reference	22	35	3	0	1	61 (26.6)
Self-surveillance/diagnosis	13	24	1	0	3	41 (17.9)
Disease guide	20	10	7	0	2	39 (17.0)
Educational aid	7	11	2	0	0	20 (8.7)
Sunscreen/UV recommendation	7	12	0	0	0	19 (8.3)
Calculator	2	9	1	0	0	12 (5.2)
Teledermatology	1	7[a]	0	0	0	8 (3.5)
Conference	2	3	0	1	0	6 (2.6)
Journal	2	4	0	0	0	6 (2.6)
Photograph storage/sharing	1	4	0	0	0	5 (2.2)
Dermoscopy	0	2	0	0	0	2 (0.9)
Pathology	0	2	0	0	0	2 (0.9)
Other	1	6	0	1	0	8 (3.5)
Total applications, No. (%)	78 (34.1)	129 (56.3)	14 (6.1)	2 (0.1)	6 (2.6)	229 (100.0)

[a] Two additional Apple iOS teledermatology applications were identified in a query of the term *teledermatology* during March 2013: HIV-Derm Algo Study (launched October 20, 2012) and SFGH Teledermatology pilot (launched November 12, 2012).

Figure 2.3: Total Applications, Brewer 2013

Using the same search criteria and categories from the article above indicates that the availability of dermatological apps is growing. Today there are 33 dermatological apps on the Apple platform that can be identified as having ”Self-surveillance/diagnostic” features. With the exception of the ”Reference” category, all other categories show significantly higher numbers of available apps.

Category	Apple 2013	Apple 2016	Android 2013	Android 2016
Reference	35	29	22	31
Self-surveillance/diagnosis	24	33	13	23
Disease guide	10	24	20	46
Educational aid	11	23	7	24
Sunscreen/UV recommendation	12	20	7	19
Calculator	9	4	2	10
Teledermatology	7	19	1	12
Conference	3	11	2	17
Journal	4	19	2	10
Photograph storage/sharing	4	3	1	1
Dermoscopy	2	7	0	3
Pathology	2	0	0	0
Other	6	8	1	4
Total	129	200	78	200

Figure 2.4: Demotological Apps by Category, July 2013(Brewer 2013) vs May 2016

It is important to note that the categories listed above are not defined in the stores. The apps must be manually assigned to a category based on an interpretation of the description text in the store and information obtainable on related websites. It is possible therefore that apps that were originally designated to the “Reference” category

might have been interpreted in this paper as a “Educational aid” app for example. The interpretation of the functionality of an app is fuzzy in many cases, and many apps have some crossover functionality. An app developed for self-surveillance will often contain information pertaining to symptoms and treatment (reference).

2.3.4 Dermatological Apps with Automatic Risk Assesment

Of the 55 apps identified belonging to the ”Self-surveillance/diagnosis” category, only 2 provided risk assesment features based on automatic analysis of captured images.

- SkinVision : https://skinvision.com
- mSkin Doctor https://play.google.com/store/apps/details?id=com.maleemtaufiq.mSkinDoctor

Chapter 3

Image Data Sources

3.1 Dermofit

High quality clinical images with corresponding masks. Great for training and testing.

Image Type	Description	Amount	Comment
Actinic Keratosis	Pre-cancerous patches of flakey or crusty skin, can develop into Squamous Cell Carcinoma	45	Not useful as comparison against melanoma
Basal Cell Carcinoma	Abnormal, uncontrolled growths of the skin's basal cells.	239	Not useful as comparison against melanoma
Dermatofibroma	Common and benign skin tumour.	65	Useful as comparison, some extreme cases might have to be left out of the training set.
Haemangioma	A collection of small blood vessels that form a lump under the skin.	97	Useful as comparison, some extreme cases might have to be left out of the training set
Intraepithelial Carcinoma	A type of squamous cell skin cancer limited to the upper layer of the skin.	97	Not useful
Malignant Melanoma	A type of cancer that develops from the pigment-containing cells known as melanocytes.	76	This is our baseline set, half will be used for training, the other half for testing
Melanocytic Nevus	Typical mole, benign.	331	Very usefull as comparison to Melanoma
Pyogenic Granuloma	Common skin growth, small, round and red in color due to large number of blood vessels.	24	Not usefull
Seborrhoeic Keratosis	Common non-cancerous skin growth.	257	Maybe usefull
Squamous Cell Carcinoma	Abnormal and uncontrolled growth of squamous cells in the epidermis.	88	Not usefull

Table 3.1: Dermofit Image Categories

3.2 DermQuest

Online teaching and learning resource with large image database.

Image were selected based on their usefullness for this project. Usefullness is based on quality and type of image. Dermoscopic images were not selected. Instead images were taken that appeared to be taken with a standard camera. The images only contained the mole to be examined and the surrounding skin area. No other details like eyelids, ears, or dark shadows.

Subgroups of Melanocytic Nevus were chosen. Dysplastic and Intradermal Nevus for the non-cancerous cases, and Malignant Melanoma as the cancerous cases.

Image Type	Description	Amount	Comment
Benign Keratosis		5	
Malignant Melanoma	A type of cancer that develops from the pigment-containing cells known as melanocytes.	39	This is our baseline set, half will be used for training, the other half for testing
Melanocytic Nevus	Typical mole, benign.	51	Very usefull as comparison to Melanoma

Table 3.2: DermQuest Image Categories

3.3 PH2Dataset

Demascopic Images

Many of the mole images extend almost to or even beyond the image border. This makes some of the processing difficult. However the images include an excel spreadsheet which clearly designates what type of mole, including some scoring info such as Asymmetry and Color information. Includes border mask images.

Image Type	Description	Amount	Comment
Common Nevus		79	
Atypical Nevus		79	
Melanoma		39	Baseline set for training

Table 3.3: PH2 Image Categories

Chapter 4

Image Feature Extraction

A digital image is a matrix of pixels that contain colour information, typically comprised of the three colour channels red, green, and blue. A person can look at an image and quickly make statements about it's content. For instance, someone might look at an image of a street scene and be able to easily say "This is an image of a street in a town, there is one automobile, three people and a building in this scene". A person would easily be able to draw outlines around the objects and be able to differentiate between areas in each object.

For a computer to be able to "make statements" about the contents of an image it must use algorithms to group together and differentiate between objects in an image. The grouping together and differentiating of areas in an image is known as *segmentation*. The statements that a computer can make about the content of an image are usually of statistical nature and are referred to as *features*. Before segmentation an image goes through a *preprocessing* stage in order to remove or reduce irrelevant information or noise.

This project concerns itself with the assessing the risk of a skin lesion being a malignant melanoma based on images captured using a smartphone camera. This limits the context of the image feature extraction requirements to a specific domain. The end goal is to analyse the pixels of the skin lesion in order to make a statement about it's characteristics. That is the feature extraction step. In order to do that though, the pixels that make up the skin lesion must be distinguishable from pixels that belong to the surrounding healthy skin; this is the segmentation step. Segmentation algorithms will be susceptible to noise and disturbances in the original image. The success rate of the segmentation algorithm can be increased if interference from noise and other disturbances can be reduced before segmentation.

The steps described above are illustrated below in figure 4.1.

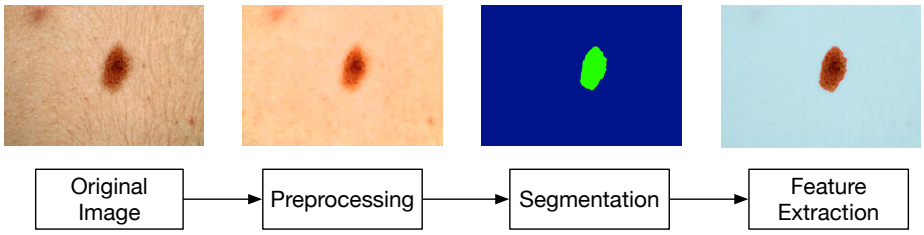

Figure 4.1: Image feature extraction steps

4.1 Preprocessing

When capturing an image using a dermatoscope the skin is illuminated equally over the skin area to be photographed. The dermatoscope captures hi-resolution images with low levels of noise. The amount of preprocessing required is limited to colour enhancement to achieve better segmentation results. Using a digital camera like those found in smartphones requires introduces other challenges [12].

Unequal illumination or shadows in the image can make it more difficult for the segmentation algorithm to precisely recognise the lesions border. Glare from too much illumination, noise introduced by the circuitry of the sensor in the digital camera or hairs on the skin can also make it difficult to differentiate between the lesion area and healthy skin.

The preprocessing stage attempts to reduce the interference cause by these factors.

4.1.1 Equalization

Histogram Equalisation in image processing is the attempt to enhance detail in images that do not utilise the full range of pixel values. Important details in an image might not be clearly visible because they are is dark areas with low contrast. A histogram of the image is calculated. The histogram shows the distribution of pixels with respect to their brightness or intensity. The intensity of the pixels can be rescaled in order to be evenly distributed.

Depending on the content of an image though global histogram equalisation often makes dark areas darker, light areas become “washed out” and noise becomes more prevalent. This the opposite of what we would like to achieve for dermatological applications. Ideally the healthy skin would be a homogeneous colour and intensity, with no noise or disturbances except for the lesion area to be extracted.

Contrast Limited Adaptive Histogram Equalisation (CLAHE) is an adaptation of Histogram Equalisation but it is not global and it limits the contrast range. CLAHE analyses small regions, or tiles, of the image and enhances the contrast so that the local histogram matches a histogram specified a “Slope” parameter. The tiles are then interpolated together to prevent edge artifacts.

Figures 4.2 to 4.4 below show 3 examples of equalised images, comparing the global histogram equalisation to the CLAHE method.

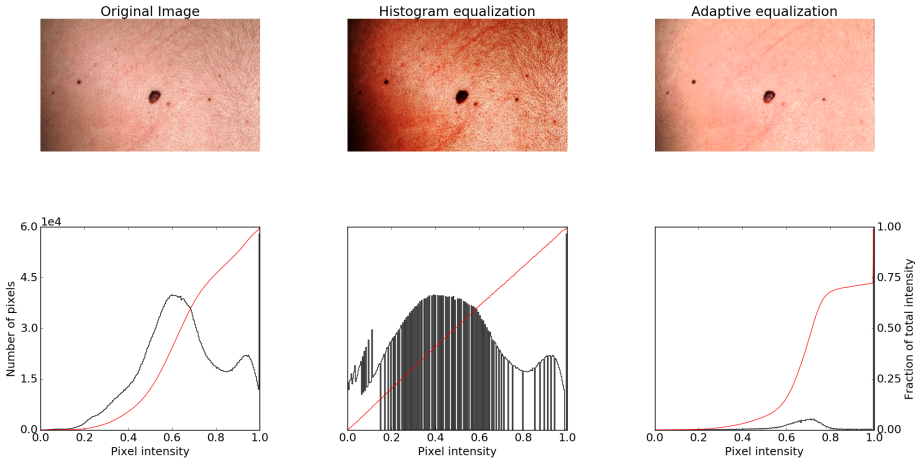

Figure 4.2: Image Equalization Example A

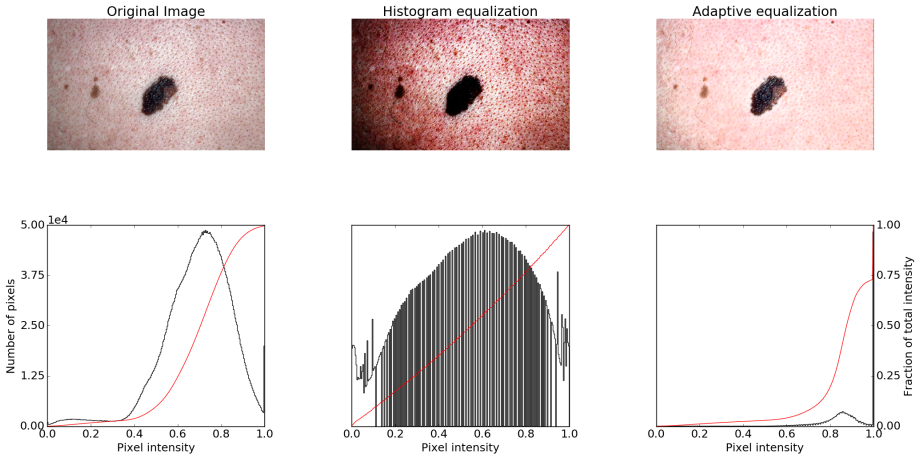

Figure 4.3: Image Equalization Example B

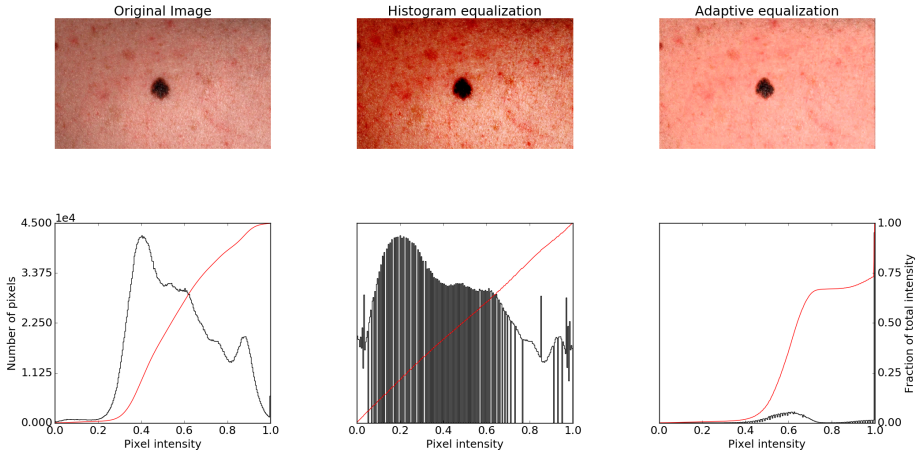

Figure 4.4: Image Equalization Example C

4.1.2 Noise Reduction

Noise in a captured image is inevitable, it will be introduced by the circuitry of the camera sensor or even from statistical quantum fluctuations of the photons known as "shot noise" [2]. Noise in the image produces outlier pixels, pixels that do not represent the surface of the object in the image, and whose values diverge from neighbouring pixels.

Noise in the image will make it more difficult for the segmentation algorithms to achieve good accurate. Other challenges though might be reflectivity of the skin's surface, the texture of the skin, or surrounding or intersecting the skin lesion.

Typical noise reduction techniques will analyse sections (often referred to as "windows" or "tiles") or the image at a time and employ some sort of averaging function to the pixel values. The gaussian and median filters are common noise reduction functions.

4.1.2.1 Gaussian Filter

The Gaussian Filter is a low pass filter that reduces high frequency components of an image. It's main parameter is sigma which defines the width or distribution or the gaussian function. Higher sigma values cause pixels to be averaged with more of their neighbours. The visual result is a blurring of the image. The Gaussian Filter is good at removing noise but at the expense of also loosing detail.

The following figures are example of Gaussian filtered images.

Figure 4.5: Gaussian Filter Example A

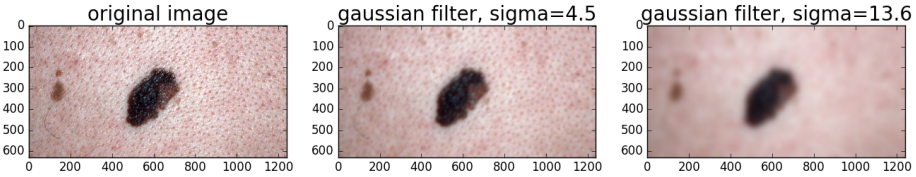

Figure 4.6: Gaussian Filter Example B

In figure 4.9 there is a hair in the lower left side of the image. At higher sigma values the hair is still partially present, while the border of the lesion has already lost much detail. The tradeoff between noise reduction and loss of important detail is not optimal for the Gaussian filter.

4.1.2.2 Median Filter

The median filter will iterate through the pixels of an image, analysing each pixel and it's immediate neighbours. The value chosen for a pixels will be the median value of the area around the pixel. In a 2D context, if the neighbourhood around a pixel is [2,1,96] the pixel will receive the value 2. The median filter is easy to compute and has the quality that edges remain well preserved. This is therefore particularly useful as a preprocessing step before segmentation.

The following figures 4.7 to **??** are examples of median filtered images.

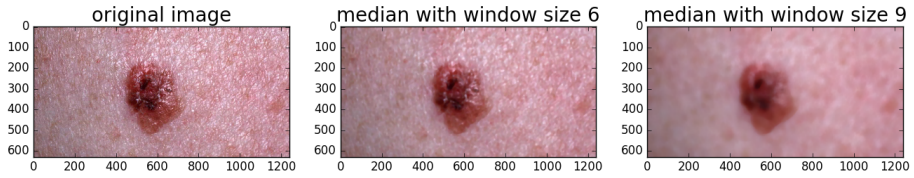

Figure 4.7: Median Filter Example A

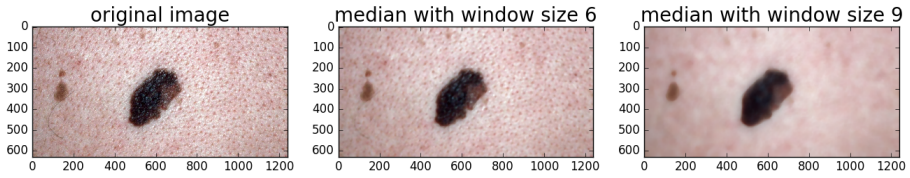

Figure 4.8: Median Filter Example B

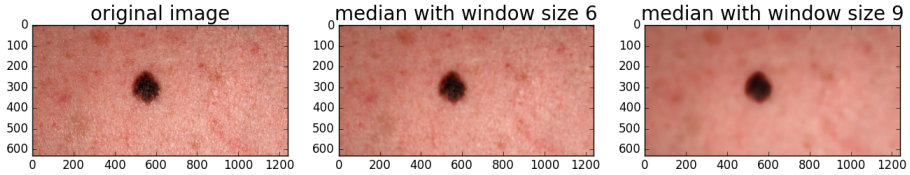

Figure 4.9: Median Filter Example C

The effect of the median window size is visible in the examples above. Larger window sizes are more effective at reducing noise and other disturbances. The edges of the lesion remain sharp, but the shape looses some detail, corners are rounded off. The size of the lesion in the image, and the image's resolution are also important factors in the success of the median filter. If the lesion is too small compared to the image size, or the resolution of the image too low, then the shape will become more distorted. This will have a negative effect on the following segmentation steps.

4.2 Segmentation

Segmentation is the process of dividing an image into distinct regions that should represent the objects in the image. For example in an application that monitors traffic congestion on a road, it would be useful count the number of cars in a video frame. To do this it's important to distinguish between groups of pixels that belong to the background and those that belong to cars.

This application concerns itself with distinguishing between pixels that belong to healthy skin surrounding a lesion, and those that belong to the lesion itself.

4.2.1 Thresholding

Thresholding describes a family of segmentation techniques that group neighbouring pixels together based on the similarity of their intensity. Basically the image is converted to a greyscale image and pixels grouped based on whether their are above or below a ratio, and if they are adjoining.

The following figures 4.10 to 4.13 are examples of segmentation using thresholding with threshold values at 39% and 50%. The regions are grouped together based on adjacency and colour coded. Before thresholding, the images were equalised using CLAHE.

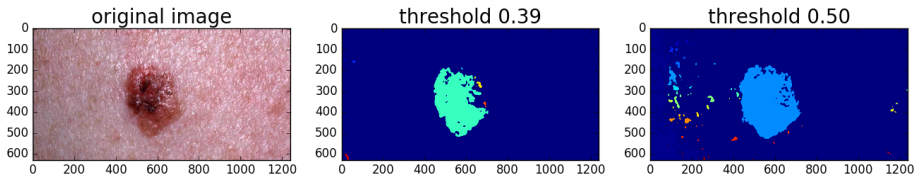

Figure 4.10: Threshold segmentation A

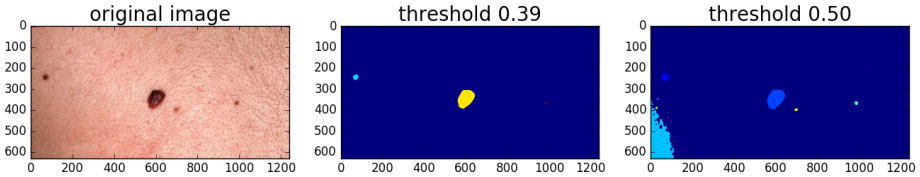

Figure 4.11: Threshold segmentation B

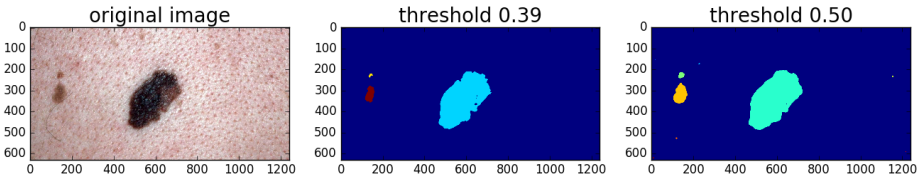

Figure 4.12: Threshold segmentation C

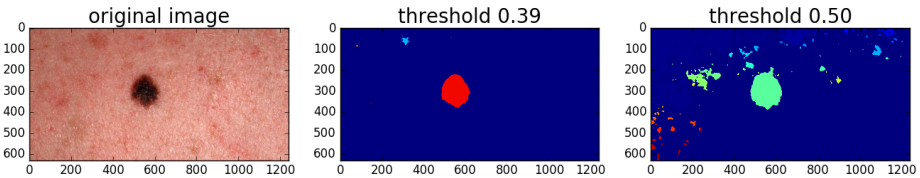

Figure 4.13: Threshold segmentation D

Depending on the content of the image and the threshold value used more regions are isolated than actually needed. It is fairly easy to discard the irrelevant regions based on some simple assumptions. We can assume that the most interesting region should be roughly in the center of the image, it should be larger than other regions, and it should not touch the edge of the image. The region remaining should be the main region of interest.

Other problems remain though. The regions of interest in figures 4.10 and 4.12 have some holes within the main region. Figure 4.13 has too much area selected at the higher threshold value, 4.10 has too little at the lower threshold setting. A static threshold setting will generally not achieve the same level of quality for all images.

By measuring the size of the main region of interest at different threshold values.

4.2.2 Color Transformation and Analysis

The thresholding method above only takes the brightness of the image into account, ignoring any colour variations. This seems counter intuitive though, especially considering the domain where one of the qualities of melanocytic nevi and melanoma is the high concentration of melanin which is responsible for skin pigmentation. It would seem natural that other image qualities such as hue and colour intensity would be important as well.

By transforming RGB images into HSV encoded image, other image qualities can be investigated more intuitively. HSV encoded images store colour information as hue (colour spectrum), saturation (colour intensity) and value (brightness).

By mapping the HSV values of an image into a 3d coordinate system where the coordinates are hue, saturation, and value some insights about the pixel grouping can be investigated. Figure 4.14 shows an image of a skin lesion next to the 3d transformation of the pixels in HSV Space. Hue and colour are the horizontal axes, value is the vertical axis.

Once can observe that the most of the pixels that make up the lesion area form a distinct clump that is connected to a larger clump that corresponds to the surrounding healthy skin.

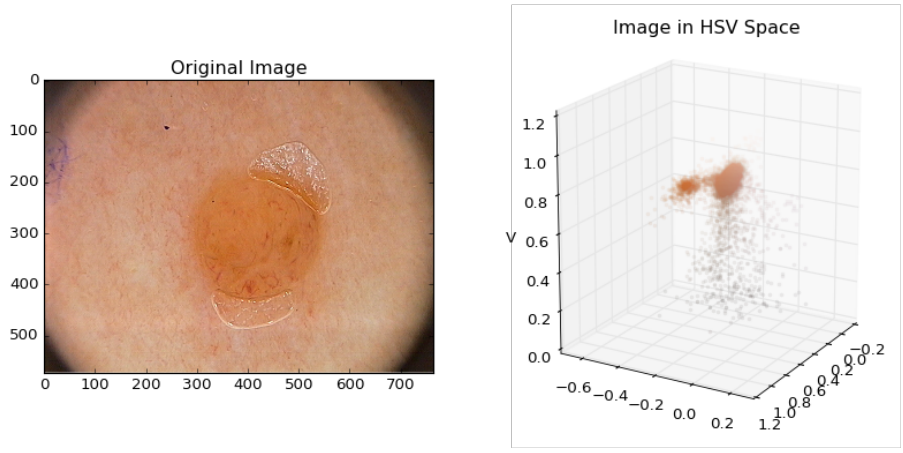

Figure 4.14: HSV Space

In Figure 4.15 the output of an application is displayed that highlights pixels in the original image that correspond to selected pixels in 3d HSV space.

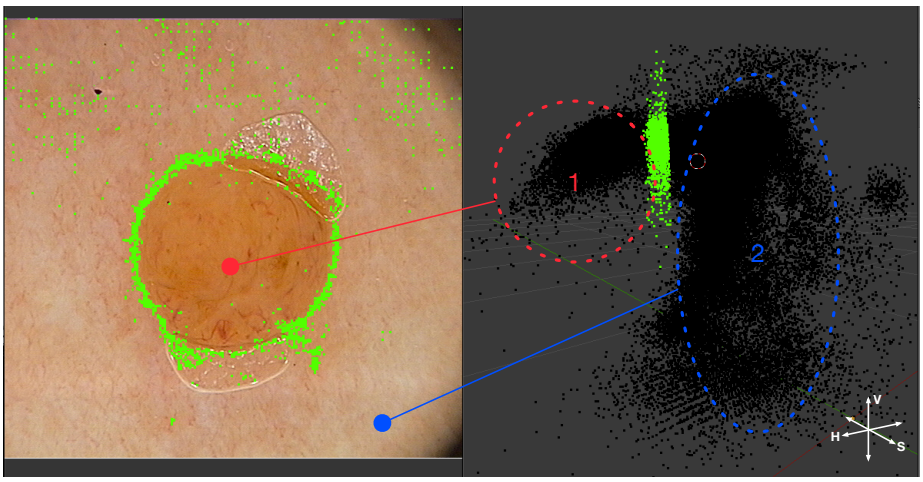

Figure 4.15: HSV Clustering visualization

This application was implemented as a python script in the Blender 3d application. The script would allow an image to be loaded into memory. The pixels would be read pixel by pixel and a point corresponding to each pixel's HSV values would be positioned in a 3d coordinate system. Each point would hold a reference to it's corresponding pixel position in the original image.

Selecting the points in the 3d view would highlight the corresponding pixel in the original image. It is clearly visible that the points lying between the 2 main clusters (1 and 2 in Figure 4.15) correspond strongly with the pixels on the border between the skin lesion and the healthy surround skin area.

Some time was spent investigating methods of grouping pixels based on both their distance in HSV space and the distance in the image space but quickly put aside because of the time and effort considerations.

The research did lead to the discovery of an already available method called Simple Linear Iterative Clustering.

4.2.3 SLIC Superpixels

SLIC is a segementaion algorithm that produces Superpixels, large groups of pixels with similar content. The edges of the Superpixel shoudld correspond to edges of objects in an image.

The SLIC algorithm uses the 3 color and 2 dimensions to cluster pixels [5]. In an iterative and computationally efficient algorithm the image is first segmented into tiles. In each iteration the borders of the tiles are recalculated based on the similarity of the pixels in the surrounding area.

Figure 4.16 illustrates the first 6 iterations of the SLIC algorithm on a skin lesion image.

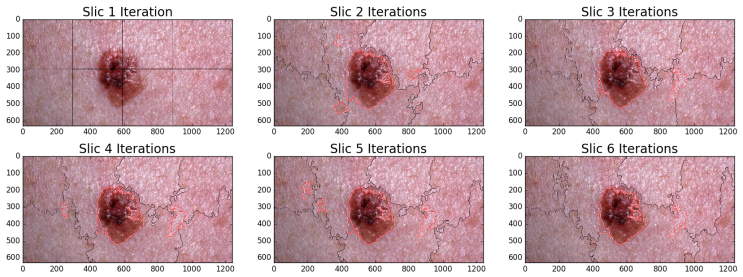

Figure 4.16: SLIC superpixel segmentation

Chapter 5

TDS Algorithm

5.1 Description of the Algorithm

Early detection of melanoma greatly increases the chances of successful treatment. A biopsy can be performed in order to gain a definitive diagnosis. However, biopsies are invasive, painful and take time. There are also visual markers Dermatologists look for in order to make a risk assessment. The ABCD Rule, also known as Stokes or TDS Calculation, looks for 4 sets of features. Based on the features the Total Dermoscopy Score (TDS) is calculated.

The 4 sets of features are Asymmetry (A), Border Irregularity (B), Color (C) and Differential Structure or Diameter (D).

Asymmetry can have a value of 0, 1 or 2 depending on the symmetry of the lesion. Where 0 is symmetric and 2 is asymmetric. A value of 1 indicates at least one axis was found across which symmetry exists. The Border score is an integer value from 0 to 8 indicating the presence of border irregularities in 8 regions. Color is an integer value from 1 to 6 indicating the presence of one to six specific colors. Similarly, the value for D indicates the presence of one to five distinct structures or textures. Alternatively, in some literature [16] D is defined as Diameter, where a diameter greater than 6mm results in a value of 5, otherwise 1.

The final TDS Score is the weighted sum of the ABCD Values and is in the range 1.0 to 8.9.

$$TDS = A * 1.3 + B * 0.1 + C * 0.5 + D * 0.5 \tag{5.1}$$

A diagnosis can be made based on the TDS Score according to the following table:

Evaluation	TDS Score
Benign	$<$4.75
Suspicious	4.75 to 5.45
Malignant	$>$5.45

Table 5.1: TDS Evaluation [19]

5.1.1 Adaptation for use in a smart phone application

The ABCD Rule, also known as the Total ***Dermoscopy*** Score, is really only applicable to images captured using a dermatoscope. Especially the differential structures are only visible on dermatoscopic images where the subsurface structures are made visible [6].

Since the D component is not applicable to images captured with a smart phone this project will use a modified TDS without the D component. The original TDS formula 5.1 can have values ranging from 1 to 8.9 and D can be 0.5 to 2.5. Without D the TDS can have values between 0.5 and 6.4.

$$TDS_{mod} = A * 1.3 + B * 0.1 + C * 0.5 \tag{5.2}$$

This results in a bening cuttoff score of 3.2 and malignagnt cuttoff of 3.7.

Evaluation	Score
Benign	$<$3.20
Suspicious	3.20 to 3.7
Malignant	$>$3.7

Table 5.2: Adapted TDS Evaluation

5.2 Automatic Calculation of ABCD Values

The ABCD Rule is used by Dermatologists to differentiate benign from malignant melanocytic tumors. It is a clearly defined rule based on easily recognizable visual features. Clinicians with limited dermoscopy experience achieve better results using the ABCD rule than other methods [19]. The following sections detail methods to automate the calculation of the ABCD scores.

5.2.1 Asymmetry

The ”Center of mass” method of measuring asymmetry is relatively easy to calculate and is more accurate compared to other complex algorithms [15].

This algorithm begins by calculating an array of radii from the lesion's center of mass to border for each of 360 degrees. The coordinates for the center of mass are calculated from the sum of all x and y coordinates of pixels within the lesion's border each divided by the sum of pixels.

For each of the 360 radii r_i a score is calculated by comparing the lengths of pairs of radii that are symmetric across r_i. If the lengths of the pair of symmetric radii have a difference of less than 10% then a point is given. The sum of points is the SFA_i (Score For Axis) for r_i.

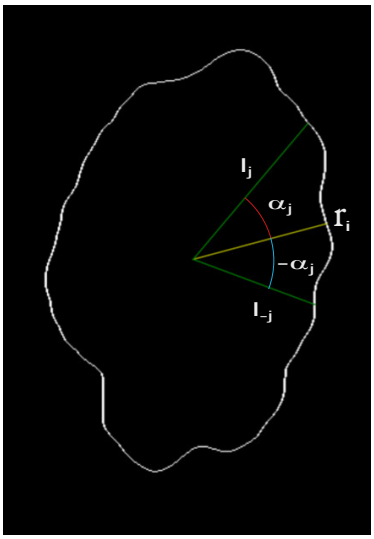

Figure 5.1: Calculate SFA for r_i

The radius with the maximum SFA score is defined as the major axis of symmetry. The SFA of the major axis as well as the perpendicular are stored. The Asymmetry score is evaluated as follows:

SFA Results	Description	Asymmetry Score
major axis $\geq$ 140 minor axis $\geq$ 140	Symmetric across both axis	0
major axis $\geq$ 140 minor axis $<$ 140	Symmetric across one axis	1
major axis $<$140 minor axis $<$140	Asymmetric	2

Table 5.3: TDS Evaluation [19]

5.2.2 Border

An image is generated where only the pixels at the border or the lesion are visible, the rest of the image is black. The pixels are gathered sequentially into an array starting at any arbitrary border pixel and traversing the border continually. It's important that the sequential order of the pixels be maintained since the change in angle and distance to the lesions center will be measured.

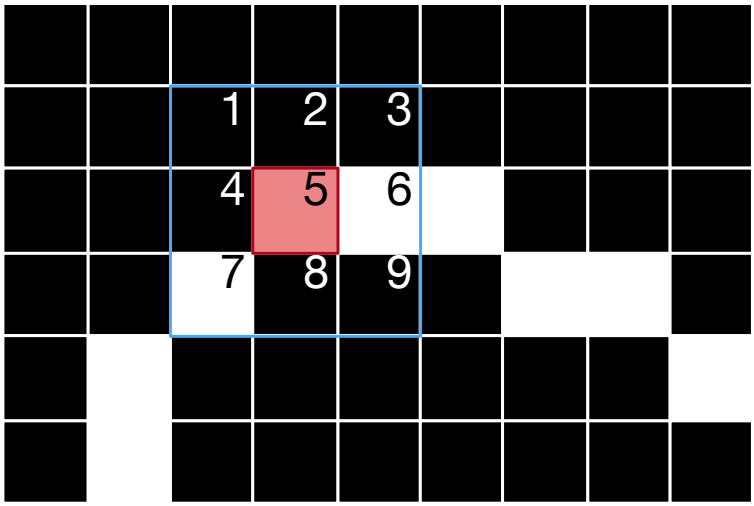

Figure 5.2: Traverse the border sequentially

An algorithm was developed that finds an arbitrary border pixel in an image that is otherwise black by iterating though pixels starting at the top left corner. When it has found a pixel that is not black it tests neighboring pixels, starting with the upper left neighbor and traversing the pixels row by row starting from the left most pixel in a row, as illustrated by the numbered pixels in figure 5.2. The center pixel, pixel number 5, is skipped.

When a non black pixel is detected it is compared to a a list or previously detected pixels. If it has not been previously detected, it is added to the list and becomes the new center. The algorithm is repeated.

If no new non-black pixel is detected the outter range is expanded by one pixel at each edge and the pixels at the edge of a 5 x 5 area are tested. This is repeated until a new pixel is found. This prevents the algorithm from halting if there are discontinuities in the border.

When the algorithm reaches the starting pixels it ends. The list of pixels now contains a sequential list of pixels that are in sequential order.

For each pixel the distance and angle from the lesion's center of mass is calculated. If a lesion's border is irregular the measure of distance and angle from the center will be erratic, whereas is the border is relatively circular and smooth the measure of distance and the difference in angle will not change abruptly between neighboring pixels.

The distance function is averaged around the distance of the start pixel (set to zero) in order to reduce discontinueities. A gaussian lowpass filter is used to reduce noise and aliasing artifacts due to pixelization.

From the smoothed distance function the first derivative is calucalated. From the angle function the difference from neighboring pixels is calculated. The resulting two functions are split into 8 equal segments. Each segment is analysed for strong variations in distance and sign changes in the difference of the angles. A sign change would

indicate a very strong change in direction of the border function. For each segment in which strong variations are detected a point in the B score is added.

A lesion with strong variations in one segment, but otherwise smooth border function would have a B score of 1, whereas a lesion with variation is each of the 8 segments would have a B score of 8.

In the figures 5.3 to 5.5 yellow rays indicate an anlge sign change, and red rays indicate strong variations in distance. The 8 white rays indicate the boundary of the 8 segments.

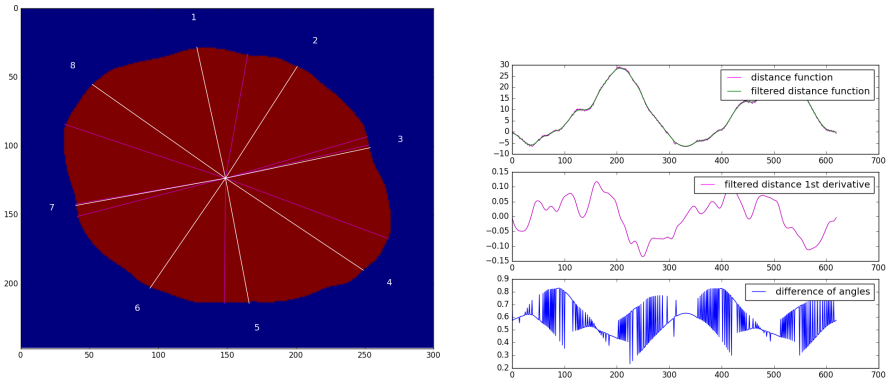

Figure 5.3: Example of border score 0

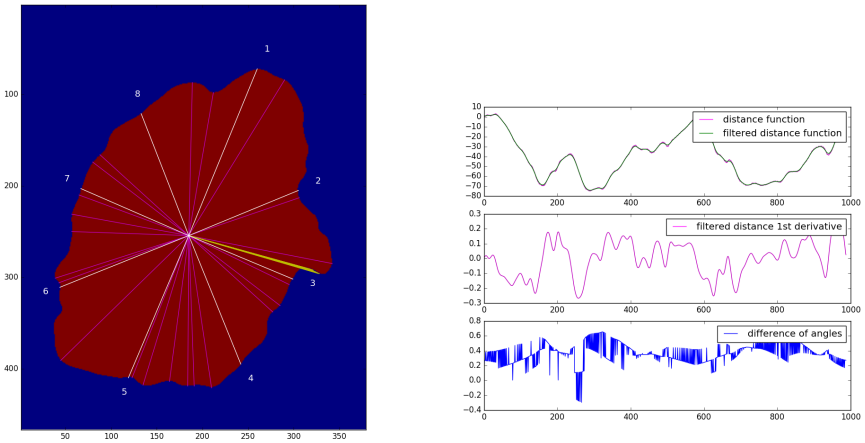

Figure 5.4: Example of border score 4

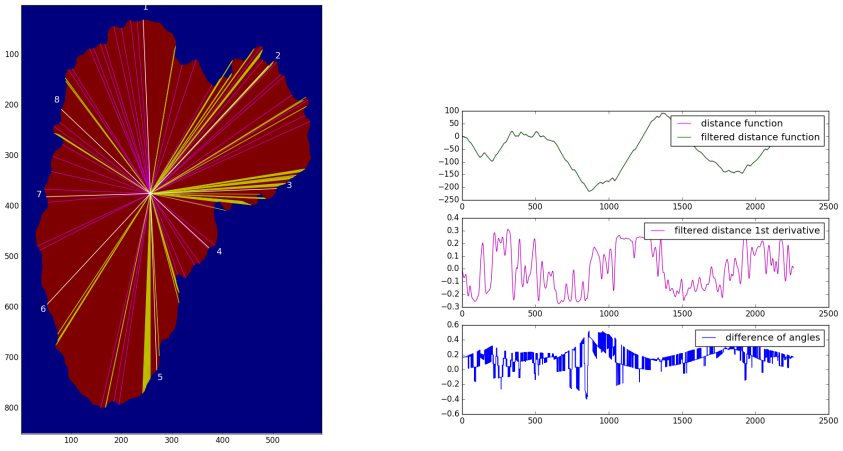

Figure 5.5: Example of border score 8

5.2.3 Color

In order to calculate the color score the *Quantification of Color* [6] algorithm was implemented in which the distance of each pixel to a defined set of colors is evaluated.

The original image is first filtered using a low pass Gaussian filter in order to reduce noise. Then for for each pixel with the lesion area the euclidean distance from 6 RGB values is calculated. The 6 RGB values, as shown in table 5.4, represent the colors that are relevent to the TDS calculation.

Color	R	G	B
White	255	255	255
Red	204	51	51
Light Brown	153	102	0
Dark Brown	51	0	0
Blue Gray	51	153	255
Black	0	0	0

Table 5.4: RGB values of the six possible TDS-relevent colors [6]

For each color a counter is incremented when a pixel is found that is closest to it. The end results are divided by the total number of pixels within the lesion's boundaries. For each color with a value greater than 0.01 a point is given to the TDS C score.

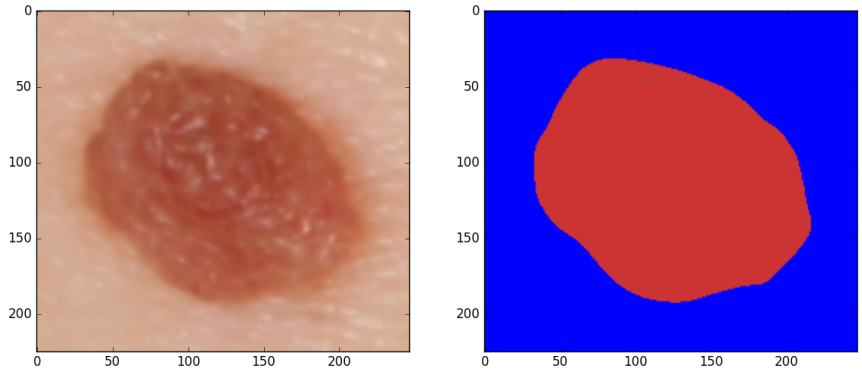

Figure 5.6: Example of color score 1

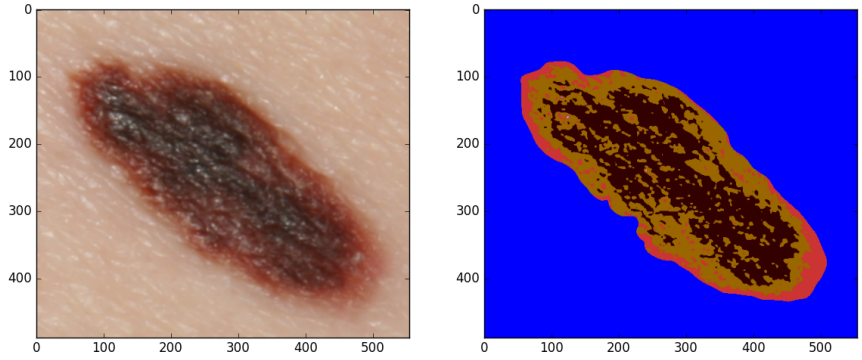

Figure 5.7: Example of color score 3

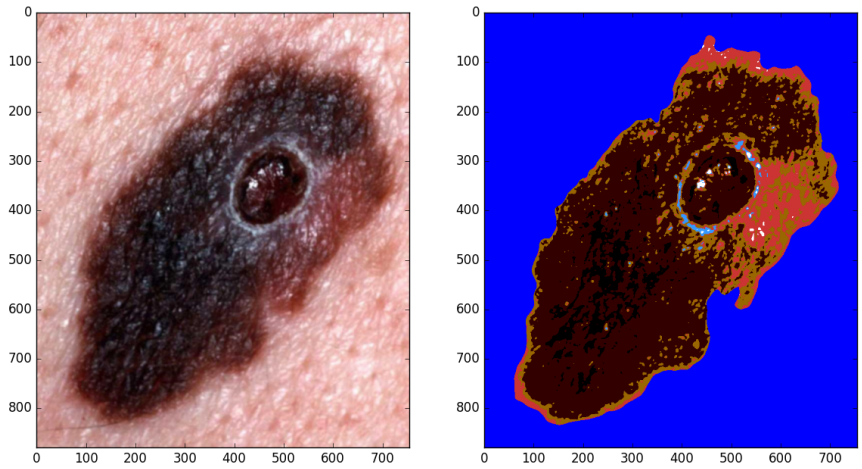

Figure 5.8: Example of color score 4

5.3 Results of Algorithm

The results of the algorithm unfortunatale are not great. It is difficult to judge though whether this is due to the algorithm itself or the quality or the border extraction algorithm on the images provied. The PH2 image set should not be considered in the results becuse the images do not really provide an accurate reflextion of images that would be caputed by a mobile camera. The Dermquest image are a better fit for this, but the selection is much smaller.

Source	False Positives	False Negatives	Correct	Total
Dermofit	10	8	115	133
Dermquest	10	4	32	46
PH2	13	0	29	42

Table 5.5: Results of TDS calculation

Source	Malignagt	Malignant Correct	Total
Dermofit	16	8	133
Dermquest	16	12	46
PH2	0	0	42

Table 5.6: Correctly Diagnosed Malignant Melanoma Images

Chapter 6

Mobile App Requirements

This chapter will describe the requirements of the application as a whole. The structure of this chapter follows the SRS Template as proposed by Wiegers in Chapter 11 and Appendix D of [20].

Table 6.1 describes the idendifiers used to reference the requirement attributes descri

ID	Attribute	Description
FE-#	Major Features	High level description of what the MDApp is expected to achieve.
LI-#	Limitations and Exclusions	Things that should not be considered part of the scope of the application
UC-#	Use Cases	Description of an activity that a user can perform.
OE-#	Operating Environment	Description of the environment the application will operate in.
CO-#	Design and Implementation Constraints	External technical factors that need to be taken account for.
AS-#	Assumptions and Dependencies	Other external factors or limitations that need to be considered.
UI-#	User Interfaces	Requirement that the user interface must fullfill.
SI-#	Software Interfaces	External software interfaces that the app must communicate with.
CI-#	Communication Interfaces	Description of communication interfaces.
PRI-#	Privacy	Privacy requirements.
INS-#	Installability	Installability requirements.
INT-#	Integrity	Data integrity requirements.
USE-#	Usability	Useability requirements.
POR-#	Portability	Portability requirements.
TES-#	Testability	Testability requirements.

Table 6.1: List of attribute identifiers

6.1 Vision and Scope

This section describes the high levels goal of the Melanoma Detection Application (MDApp) as well as to define boundaries that limit what can be expected of the app.

6.1.1 Business Case

A smart phone based assistant would provide a valuable service by making it easy, painless, inexpensive, and fast to get a preliminary risk assessment of a skin lesion. It is too easy to postpone making an appointment with a dermatologist to have a suspect lesion examined. Visiting a dermatologist is not on most people's list of fun things to do, it requires time and effort. By the time a lesion is visually compelling enough, it might be too late.

A smart phone based application that could make a preliminary diagnosis in near realtime would be a great time save and offer compelling enough information to actually make the appointment with a dermatologist.

6.1.2 Scope and Limitations

This chapter describes which features are part of the scope of the Melanoma Detection Application, what can be expected of it, and what are it's limitations.

6.1.2.1 Major Features

FE-1 : The smart phone app allows the user to capture and analyse an image of a skin lesion and provide a risk assessment to the user of the lesion being a malignant melanoma.

FE-2 : The app allows the user to create, view, edit or delete metadata that is associated with an image and corresponding risk assessment.

FE-3 : The app allows the user to save or archive the image and corresponding assessment for future comparison and review.

FE-4 : The user can browse archived images, assessments, and associated metadata.

FE-5 : The user can send a set of images with associated assessment and metadata via email.

FE-6 : The app will run on all popular smart phone devices and systems.

FE-7 : The analysis algorithms employed by the app should be easily updatable and extendable.

The partial feature tree in figure 6.3 is a visual representation of some features grouped heirarchically. First level main features are in boxes. Branching off from the first level features, are the second and third level features.

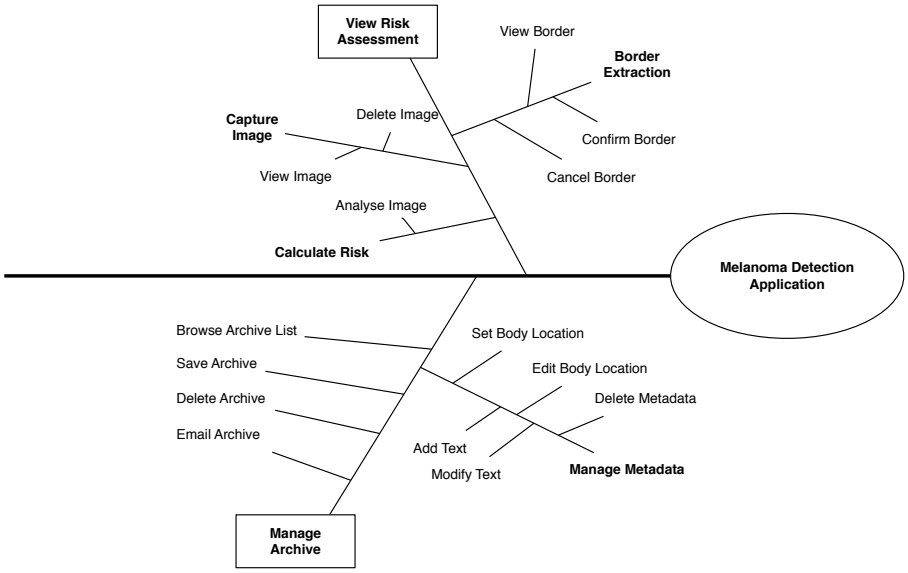

Figure 6.1: Partial feature tree for the Melanoma Detection App

6.1.2.2 Limitations and Exclusions

LI-1 : The Melanoma Detection App will use visual indicators to calculate the risk of a lesion being a dangerour melanoma. Other types of skin cancer such as basal call carcinoma or squanous cell carcinoma have different characteristics making them less easy to visually detect. The Melanoma Detection App will not take any other types of skin cancer or skin conditions into acount when calculating the risk assessment for melanoma.

LI-2 : Typical Smartphone camera's might not have the ability to capture images at close enough range needed to capture a skin lesion's details at high enough resolution. It might be necessary for the Smartphone User to additionaly purchase a readily available clip-on macro lens for smart phones.

6.1.3 Use Cases

10 Important Use Cases have been identified. The schema as defined in Chapter 8 and Appendix D of [20] was used to capture the attributes of each Use Case. The Use Case UML diagram in figure 6.2 illustrates the most important use cases. Tables 6.2 to 6.11 describe the use cases in detail.

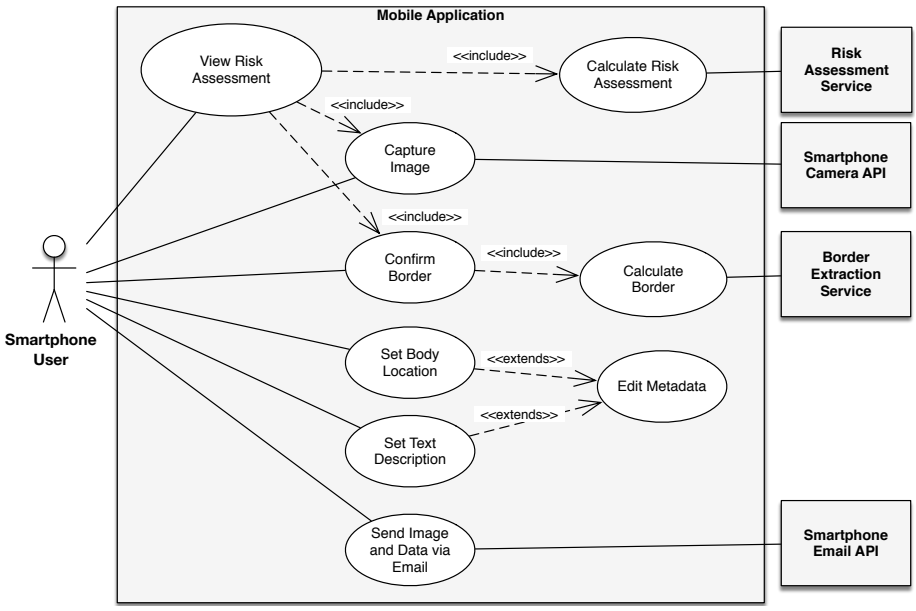

Figure 6.2: Use Case UML diagram

ID and Name:	**UC-1: Capture Image**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone Camera System
Description:	The User is shown a real time preview of the camera's input. With the help of visual indicators layered over the real time preview the User can optimise the distance and angle of the Smartphone. When the real time preview appears to show an optimal image the User can activate the image capture process.		
Trigger:	The User has navigated to the Image Capture view of the App.		
Preconditions:	None		
Postconditions:	POST - 1 : Image file is saved on the Smartphone file storage system POST - 2 : A reference to the image is saved in the MDApp		
Normal Flow:	**1.0 Capture an Image** 1. User is presented with a real time preview of the camera's input. 2. User adjusts the camera position until the real time preview is optimal. 3. The User activates the image capture process 4. The MDApp acquires the image from the Smartphone Camera System 5. The MDApp stores the image and a reference		
Alternative Flow:	None		
Exceptions:	1.0.E1 : Camera hardware is not available		
Priority:	High		

Table 6.2: UC-1

ID and Name:	**UC-2: Confirm Border**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Border Extraction Service
Description:	The MDApp initiates the border extraction calculation. When the calculation is complete the User is presented with a visual rendering of the extracted border. The MDApp prompts the user to confirm or reject that the border was calculated precisely.		
Trigger:	The User captured an image.		
Preconditions:	PRE - 1 MDApp saved and image and a reference		
Postconditions:	POST - 1: A datafile of the extracted border is saved to the Smartphone's file storage system. POST - 2 : A reference to the Datafile is saved in the MDApp		
Normal Flow:	**1.0 Confirm Border** 1. The MDApp automatically initiates the border extraction process. 2. When the border extraction process is finished the User is presented with a visual rendering of the border over the image of the skin lesion for comparison. 3. The MDApp prompts the User to confirm that the border precisely outlines the area of the lesion. 4. If confirmed, the MDApp stores the border datafile and a reference.		
Alternative Flow:	**1.3 Reject Border** 3. The User rejects the border because is does not precisely outline the lesion area. 4. The User is returned to step 1 of UC-1		
Exceptions:	2.0.E1 : border calculation service is not responding. 2.0.E2 : a border could not successfully be calculated.		
Priority:	High		

Table 6.3: UC-2

ID and Name:	**UC-3: View Risk Assessment**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Risk Assessment Service
Description:	The MDApp initiates the risk assessment calculation. When the calculation is complete the User is presented with the results.		
Trigger:	The Images Border was confirmed by the Smartphone User.		
Preconditions:	PRE - 1: A datafile of the extracted border was confirmed by the Smartphone User		
Postconditions:	POST - 1: A risk assessment data for the captured image is available.		
Normal Flow:	**1.0 Confirm Border** 1. The MDApp automatically initiates the risk calculation process. 2. When the risk assessment calculation is finished the User is presented with the results.		
Exceptions:	3.0.E1 : risk assessment service is not responding. 3.0.E2 : a risk assessment could not successfully be calculated.		
Priority:	High		

Table 6.4: UC-3

ID and Name:	**UC-4: Set Body Location**		
Primary Actor:	Smartphone User	**Secondary Actor:**	
Description:	The User is presented with a visual representation of a human body. The User selects a location on the body. The MDApp saves the body location data as metadata associated with captured image.		
Trigger:	The User navigates to the metadata view of the MDApp		
Preconditions:	PRE - 1 MDApp saved and image and a reference		
Postconditions:	POST - 1: Metadata describing the location of the lesion is stored with a reference to the image of the lesion.		
Normal Flow:	**1.0 Set Body Location** 1. The user navigates the the metadata view of the MDApp. 2. The MDApp presents the user with a visual representation of the human body. 3. The User selects an area of the body. 4. The MDApp stores the body location as metadata associated with the captured image.		
Priority:	Medium		

Table 6.5: UC-4

ID and Name:	**UC-5: Set Text Metadata**		
Primary Actor:	Smartphone User	**Secondary Actor:**	
Description:	The User is presented with text area where she can enter any text that should be associated with the captured image		
Trigger:	The User navigates to the metadata view of the MDApp and scrolls to the text area.		
Preconditions:	PRE - 1 MDApp saved an image and a reference		
Postconditions:	POST - 1: Metadata test is stored with a reference to the image of the lesion.		
Normal Flow:	**1.0 Set Text Metadata** 1. The MDApp presents the user with a text area 2. The User enters text in the text area 3. The MDApp stores the text as metadata associated with the captured image		
Priority:	Medium		

Table 6.6: UC-5

ID and Name:	**UC-6: Save Archive**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone File System
Description:	The User may store an image and associated metadata for review and comparison at a later date.		
Trigger:	The User clicks the “Save” button		
Preconditions:	PRE - 1 MDApp saved an image and it’s reference.		
Postconditions:	POST - 1 The MDApp saves the image and associated metadata as an archive to the Smartphone’s file system.		
Normal Flow:	**1.0 Save Archive** 1. The User clicks the “Save” button. 2. The MDApp gathers last captured image together with the associated metadata (including the calculated border and risk assessment if available) and saves this to the Smartphone File System.		
Priority:	Medium		

Table 6.7: UC-6

ID and Name:	**UC-7: Browse Archive List**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone File System
Description:	The User may scroll through a list of previously archived images and metadata. The list provides a preview or the image as well as the datae of when the image was captured. The list is ordered by the date of capture.		
Trigger:	The User navigates to the archive-list view.		
Preconditions:	PRE - 1 : At lease one image as previously been archived.		
Postconditions:	None		
Normal Flow:	**1.0 Browse Archive List** 1. The MDApp presents the user with a navigable list of previously saved archived images. 2. The User can navigate or scroll through the list.		
Priority:	Medium		

Table 6.8: UC-7

ID and Name:	**UC-8: View Archive Detail**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone File System
Description:	The User may select an archive from the archive list and the MDApp with provide the user with a detail view of the selected archive.		
Trigger:	The User selects an archive from the archive-list view.		
Preconditions:	PRE - 1 : At lease one image as previously been archived.		
Postconditions:	POS - 1 : A detail view of a previously archived image and associated metadata to visible to the user. POS - 2 : The selected archive is designated by the MDApp as the active archive.		
Normal Flow:	**1.0 View Archive Detail** 1. The User selects an item in the list of archived images 2. The MDApp designates the selected item as the active archive. 3. The MDApp presents a view showing image and associated metadata to the user.		
Priority:	Medium		

Table 6.9: UC-8

ID and Name:	**UC-9: Delete Archive**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone File System
Description:	The User may delete an image and associated metadata from the list of archives. This will remove the image and associated metadata from the device storage.		
Trigger:	The User clicks the “Delete” button.		
Preconditions:	PRE - 1 : An archive exists that is the current active archive. PRE - 2 : The User is in the Archive Detail View.		
Postconditions:	POST - 1 : The previously active archive is deleted from the Smartphone’s file system and is no longer listed in the archive-list. POST - 2 : The current active archive is set to NULL.		
Normal Flow:	**1.0 Delete Archive** 1. The user clicks the ‘delete’ button. 2. The MDApp provides a confirmation-view with a “confirm” button, 3. The user clicks the “confirm” button. 4. The MDApp requests the smartphone’s file system to delete the images and associated metadata that comprised the current active archive. 5. The current active archive is set to NULL		
Alternative Flow:	**1.3 Do not confirm** 3. The user clicks the “cancel” button. 4. The user is brought back to the precondition state.		
Priority:	Medium		

Table 6.10: UC-9

ID and Name:	**UC-10: Email Archive**		
Primary Actor:	Smartphone User	**Secondary Actor:**	Smartphone Email System
Description:	The User may send an archived image and associated metadata by email to a dermatologist or doctor for review. A subject and message text can be added by the user before submitting via one of the email accounts registered in the mobile phone's system.		
Trigger:	The User touches the "send" icon.		
Preconditions:	PRE - 1 : The Smartphone OS is configured with a valid email account. PRE - 2 : An archive exists that is the current active archive. PRE - 3 : The User is in the Archive Detail View.		
Postconditions:	None		
Normal Flow:	**1.0 Email Archive** 1. The user clicks the "send" button. 2. The MDApp provides the user with the smartphone systems standard email composer. 3. The MDApp sends a request to the smartphone's email system with the archive data as an attachment. 4. The MDA		
Priority:	Medium		

Table 6.11: UC-10

6.2 Software Requirements Specification

6.2.1 Introduction

6.2.1.1 Purpose

The Software Requirements Specification will describe the functional and nonfunctional requirements of the Melanoma Detection App. It is meant to be a guideline for developers who will be implementing the mobile application.

6.2.2 Overall Description

6.2.2.1 Product Perspective

The Melanoma Detect App (MDApp) will provide the user with a risk assessment of a skin lesion. The App will guide the user through the process of capturing an image and confirming the correct recognition of the boundary of the lesion. Once confirmed the App will analyse the visual features of the lesion based on dermatological rules and provide the user with an initial risk assessment.

The App should help motivate the user to make an appointment with a dermatologist when a significant risk has been detected. It is not meant to replace the need to visit a dermatologist though.

In order to provide these services the MDApp must be able to utilise the features and capabilities of the smart phone that it runs on. The Partial Context Diagram in fig-

ure 6.3 shows the MDApp in relation to the external components that it must interface with. Email, camera, and file system are system components within the smart phone that the app must communicate with. The border extraction and risk assessment services are online services that the MDApp must send image data to in order to receive the border data and risk assessment. Finally, to make it all work together, the MDApp must provide an interface to the user that is clear and unobtrusive, guiding the user through all the necessary steps.

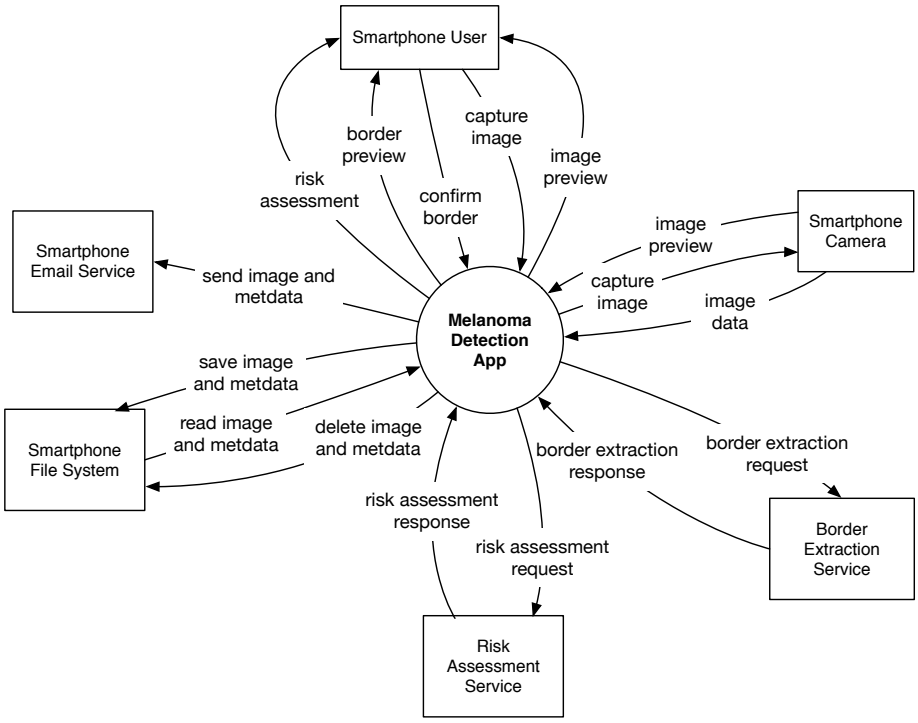

Figure 6.3: Context Diagram of the Melanoma Detection App

6.2.2.2 User Classes and Characteristics

The Melanoma Detection App only has one class of user, the Smartphone User. The Smartphone User has one or several skin lesions that she or he is worried about. The Smartphone User is not expected to have any technical or dermatological expertise.

6.2.2.3 Operating Environment

OE-1 : The Mobile App shall operate on the following platforms: Android OS (minumum v.5.2.2) and iOS (minumum v.9.3)

6.2.2.4 Design and Implementation Constraints

CO-1 : The Border Extraction and Risk Assessment Services shall be implemented on a server in oder to leverage the effort already made with python and python based libraries for image processing and scientific calculations. Internet access is therefore a requirement for the app to run.

CO-2 : The method for archiving images and risk assessment data will use whatever standard is the most accepted for the relevant platform. i.e. iCloud on iOS.

6.2.2.5 Assumptions and Dependencies

AS-1 : It is assumed that the user of the software is also the user of the mobile phone. The software might need to request permission from the user to access the camera or network.

AS-2 : It is assumed that the user of the software is also the only user of the mobile phone. Therefore no additional measures need be taken to secure the app from other potiential users of the phone. The app will be as secure as the phone itself.

6.2.3 System Features

6.2.3.1 Capture an Image of a Skin Lesion

6.2.3.1.1 Description

The MDApp allows the Smartphone User to capture an image of a skin lesion.

6.2.3.1.2 Functional Requirements

Image.Capture:	**Capture image using camera**
.Preview:	The App must provide the User with a realtime preview of the camera input. It will include some visual indicators that help the user capture an optimal image.
.Trigger:	The User will trigger the capture when he or she decides the realtime preview is optimal.
.Response:	The App must be able to capture a still image from the camera when the User pushed the capture button. The image will be saved in a standard image format on the phone's file system.
.Display:	The App will present the image to the user on the device's screen.

6.2.3.2 Extract the Border of a Skin Lesion

6.2.3.2.1 Description

In oder to be able to properly analyse the skin lesion image, the MDApp must be able to distinguish pixels that belong to the lesion from pixels that belong to the healthy skin area surrounding the lesion.

6.2.3.2.2 Functional Requirements

Border.Extract:	**Calculate border of skin lesion**
.Calculate:	The App sends a captured image as a request to the Border Extraction Service. When the service has completed the calculation the results of the border calculation are provided by the service to the App.
.NA	
.Err	
.Display:	The App will present the results or the border calculation to the User on the device's screen.
.Promt:	The App will prompt the User to confirm that the border was precisely calculated.
.Response:	The User can confirm or cancel the border.

6.2.3.3 Calculate the Risk Assessment Skin Lesion

6.2.3.3.1 Description

The MDApp sends the skin lesion image and border data as a request to the Risk Assessment Service.

6.2.3.3.2 Functional Requirements

Risk.Assessment:	**Calculate risk assessment of lesion image**
.Calculate:	The App sends a captured image and border data as a request to the Risk Assessment Service. When the service has completed the calculation the results of the risk assessment are provided by the service to the App.

.Display:	The App will present the results or the risk assessment calculation to the User on the device's screen.

6.2.3.4 Create, Modify, and Delete Metadata

6.2.3.4.1 Description

The Smartphone User can add metadata to a captured image. This can include the location on the body that the image was captured from, date of capture, and a text description. This metadata can be associated with an image so that it can be used at a later data for comparison and review.

6.2.3.4.2 Functional Requirements

Metadata.Location:	**Specify the lesion's location on body**
.Display:	The MDApp will present an image of the human body to the Smartphone User.
.Promt:	The MDApp will prompt the Smartphone User to select or zoom in on a location on the image of the human body.
.Response:	The selected location will be stored as metadata associated with the image of the lesion.
Metadata.Text:	**Add metadata describing lesion**
.Display:	The MDApp will present a text field in which the Smartphone User may enter, edit, or delete text.
.Response:	The entered text will be stored as metadata associated with the image of the lesion.

6.2.3.5 Save, View, Edit, Delete and Email Archive

6.2.3.5.1 Description

The Smartphone User can browse a list of previously analysed and saved images (archives). From the list, an image may be selected by the Smartphone User to view in more detail. When selected the MDApp will present the saved image, border, risk assessment, and associated metadata to the Smartphone User for review. An archive from the list may also be deleted or sent as an email.

6.2.3.5.2 Functional Requirements

Archive.Save:	**Save image and metadata as archive**
.Display:	—
Archive.Browse:	**Browse list of archived images**
.Display:	The MDApp will present the Smartphone User with a list of saved archive images.
Archive.Select:	**Select archived image to view or edit**
.Display:	The MDApp will present the Smartphone User with a detail view of the selected archived image and associated metadata.
.Edit:	The MDApp will present a text field in which the Smartphone User may enter, edit, or delete text.
.Delete:	The MDApp will present a text field in which the Smartphone User may enter, edit, or delete text.
.Email:	The MDApp will present a text field in which the Smartphone User may enter, edit, or delete text.

6.2.4 Data Requirements

6.2.4.1 Logical Data Model

The entity-relationship diagram in figure 6.4 is a high level view of the main groups of information and their interconnections. The figure uses the Peter Chen notation, where the diamonds represent the relationships between pairs of entities. Cardinality of the relationships is shown as numbers on the lines that connect the relationships to the entities. The entities are the boxes. The relationships can be read for example as "Metadata describes the Lesion Image" (active voice) or "The Lesion Image is described by Metadata" (passive voice). The order of the entities in the diagram does not define the relationship, the object and subject must be inferred. For instance the relationship between Archive and Lesion Image is read "The Archive stores many Lesion Images" has the opposite positioning in the diagram compared with the relationship between Metadata and Lesion Image. The "direction" of the relationship is specified in the following data dictionary along with the entities attributes in detail.

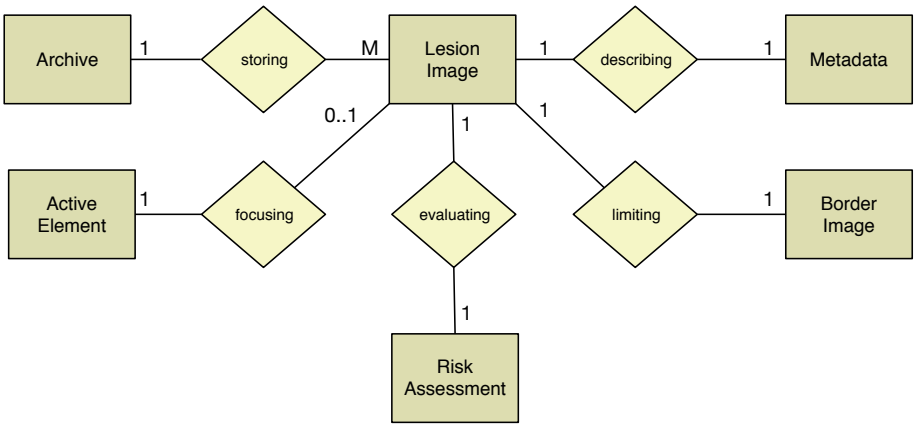

Figure 6.4: Patial entity relationship model of the Melanoma Detection App

6.2.4.2 Data Dictionary

The Data Dictionary 6.4 lists the data entities in detail. Entities can be primitives or composites. Primitives are simple data elements like an integer, float, or string. Composites are collections of entities which can themselves be primitives or composites. The entities that are part of a composite are listed in the Composition or Data Type column separated by a "+" sign. Each entity in a composite will also be listed separately and described in the data dictionary below. If applicable the length and range of possible values will be described for primitiv entities.

Data element	**Description**	**Componsition or data type**	**Length**	**Values**
Lesion Image	reference to the captured image	Image ID + File Path + Creation Date		
Image ID	unique identifier for an image	integer	11	system-generated sequential integer beginning with 1
File Path	location of a file on the file system	string	256	
Creation Date	date and time that a file was created	DDMMYYYY:HHSS		
Metadata	information describing the lesion and it's location	Lesion Image + Lesion Description + Lesion Location		
Lesion Description	User defined text that describes the lesion	alphanumeric		

Lesion Location	information describing the lesion location on the body	Lesion Image + Body Location ID + X Coordinate + Y Coordinate		
Body Location ID	id of a region on the body	integer	2	1 : Head Front 2 : Torso Front 3 : Right Upper Arm 4 : Left Upper Arm 5 : Right Forearm 6 : Left Forearm
X Coordinate	x coordinate of the lesion in the region specified by the body location id	integer	128	
Y Coordinate	y coordinate of the lesion in the region specified by the body location id	integer	128	
Border Image	data that defines the outline of the lesion area	Lesion Image + File Path + Creation Date		
Risk Assessment		Lesion Image + Assessment Date + Version Number + SFA Major + SFA Minor + Border Irregularity + Color Score + TDS Score		
Assessment Date	date and time that the risk assessment was calculated	DDMMYYYYHHSS		
Version Number	version number of the risk assessment service software	integer	4	
SFA Major	measure of symmetry of the lesions major axis	integer	3	0 - 360
SFA Minor	measure of symmetry of the lesions minor axis	integer	3	0 - 360
Border Irregularity	measure of irregulatiry of the lesion's border	integer	1	0 - 8
Color Score	number of specific colors found in the lesion's image	integer	1	0 - 8

TDS Score	weighted linear combination of the image features summarizing the risk assessment	float		1 - 8.9
Archive		Archive ID + Lesion Image + Metadata + Border + Risk Assessment + Archive Date		
Active Element	This points to the Lesion Image currently being processed and on which the graphical user interface is acting on.	Lesion Image + Metadata + Border + Risk Assessment + Archive Date		

Table 6.17: Partial data dictionary for the Melanoma Detection App

6.2.4.3 Data analysis

6.2.4.3.1 CRUD Matrix

A CRUD (Create, Read, Update, Delete) matrix can highlight which use cases interact with which data entities and in which way. Every entity that is read or deleted also needs to be created somewhere. The CRUD matrix in figure 6.5 shows that this is the case.

Entities / Use Cases	Lesion Image	Metadata	Border Image	Risk Assessment	Archive
Capture Image	C	C	C		
Confirm Border				C	
View Risk Assessment				R	
Set Body Location		U			
Set Text Metadata		U			
Save Archive					C
Browse Archive					R
View Archive	R	R	R	R	R
Delete Archive	D	D	D	D	D
Email Archive	R	R	R	R	R

Figure 6.5: CRUD matrix for Melanoma Detection App

6.2.4.4 Reports

6.2.4.4.1 Email Report

When the Smartphone User chooses to send an archive as an email, the data must be flattened into a format that is email compatible. The body of the email will be an html document that displays the lesion image, the calculated border, the risk assessment data and the associated metadata.

6.2.5 External Interface Requirements

6.2.5.1 User Interfaces

UI-1 : The UI shall guide the Smartphone User sequentially through the process of capturing and analysing an image. The Smartphone User should not be able to jump through states in wrong sequence.

UI-2 : The UI shall allow the Smartphone User to capture an image easily, using only one hand.

UI-3 : When waiting for a process to finish or a response from an online server the UI should visually indicate that it is waiting for a long process. (i.e. throbber, spinning wheel, progress indicator)

UI-4 : The UI shall allow the Smartphone User to select her or his preferred language (i.e. localisation)

UI-5 : The UI shall conform to native UI standards as much as possible.

UI-6 : Unless in conflict with UI-5 the UI shall have the same components across multiple platforms.

6.2.5.2 Software Interfaces

6.2.5.2.1 Smartphone Camera

The MDApp will interface with the Smartphone's camera hardware via the systems camera API.

SI-1.1 : The MDApp shall activate the camera when needed and be able to receive a live preview stream from the camera.

SI-1.2 : If the native API allows, the MDApp should set to camera's focus mode to Macro.

SI-1.3 : If the native API allows, the MDApp should register a call back to be informed when the camera's autoFocus has focused.

SI-1.4 : The MDApp shall signal the camera to capture the highest resolution image possible when the Smartphone User triggers an image capture event.

6.2.5.2.2 Smartphone File System

Captured images and Border Analysis images must be stored on the Smartphone File System in such a manner so that they are accessible by the Smartphone User in the native methods. The native file access, file handling, and backup methods for the respective Smartphone platform shall apply to files created and captured by the MDApp

SI-2 : The MDApp will save files to the Smartphone's native file system.

6.2.5.2.3 Smartphone Email Service

SI-3 :

6.2.5.3 Communication Interfaces

6.2.5.3.1 Border Extraction Service

The Border Extraction Service is an online web based service that the MDApp can communicate with for the purpose of extracting the precise border information from a captured image of a skin lesion.

CI-1.1 : The MDApp will upload a captured skin lesion image to the web-based Border Extraction Service. The files will be uploaded as part of a multipart/form-data html POST method.

CI-1.2 : The MDApp will receive a confirmation from the service as JSON, including an id that will be used to poll the service for progress.

CI-1.3 : The MDApp will poll the online service regularly, the service will indicated that the Border Extraction processes is in progress or has finished, in which case it will also include the url where the border image can be downloaded.

CI-1.4 : The MDApp will download the border data image from the Border Extraction Service

6.2.5.3.2 Risk Assessment Service

The Risk Assessment Service is an online web based service that the MDApp can communicate with for the purpose of evaluating the risk of a skin lesion being a melanoma.

CI-2.1 : The MDApp will upload a captured skin lesion image and border image to the web-based Risk Assessment Service. The files will be uploaded as part of a multipart/form-data html POST method.

CI-2.2 : The MDApp shall receive a confirmation from the service as JSON, including an id that will be used to poll the service for progress.

CI-2.3 : The MDApp will poll the online service regularly, the service will indicated that the Risk Assessment processes is in progress or has finished, in which case it will respond with the results of the assessment as JSON.

6.2.6 Quality Attributes

This section lists and describes the non-functional requirements.

6.2.6.1 External Quality Attributes

6.2.6.1.1 Privacy

PRI-1 : No information about the user shall be transmitted to any external services with the exception of when the user sends the image and metadata via email.

6.2.6.1.2 Installability

INS-1 : Installing the MDApp shall not require any special instructions other than those a normal Smartphone User would expect.

6.2.6.1.3 Integrity

INT-1 : Any data created by the Smartphone User through use of the MDApp shall be backup-able through whatever native backup solutions the native platform offers.

INT-2 : Data and state shall be persisted, so that when reopened, the MDApp will have the same state as when it was last closed.

INT-3 : Data and state shall be persisted, so that no data is lost when the MDApp is updated.

6.2.6.1.4 Usability

USE-1 : Every function on the MDApp can be completed with one hand.

6.2.6.2 Internal Quality Attributes

6.2.6.2.1 Portability

POR-1 : All of MDApp's system features shall be implemented across all the device platforms it runs on.

6.2.6.2.2 Testability

TES-1 : Individual components of the system must be testable in isolation from one another.

6.2.6.2.3 Modifiability

MOD-1 : The code should be grouped into logical categories that are not too inter-dependent. This attribute is not easily quantifiable, but is influenced by the software architecture and development frameworks employed. A software architecture should be chosen that fits the responsibilities of the application, and frameworks that integrate data-binding, templates, and ui composition techniques can be leveraged to reduce lines of code.

6.2.6.3 Quality Attribute Prioritisation

Depending on the domain of an application quality attributes might vary in importance. Prioritising the attributes will help in making descisions regarding the architecture and design of an application. Choosing a framework that targets achieving high performance may have a negative impact on portability, if portability has a high priority, then alternative frameworks should be considered. In figure 6.6 the quality attributes are compared pair-wise and tallied.

Attribute	Score	Weight	Privacy	Installability	Integrity	Usability	Portability	Testability	Modifiability
Privacy	2	0,095		<	^	^	<	^	^
Installability	0	0,000			^	^	^	^	^
Integrity	6	0,286				<	<	<	<
Usability	4	0,190					<	^	<
Portability	3	0,143						<	<
Testability	3	0,143							^
Modifiability	3	0,143							
Total:	21	1							

Figure 6.6: Prioritisation of Quality Attributes

Chapter 7

Architecture Design

7.1 Software Architecture

Depending on the application domain, distinct architectural patterns exist that define areas of responsibility that are used to group together software components. Choosing an architectural pattern that fits the application domain can help to organise code, reduce complexity and leverage the experience of developers that have solved similar problems in the past.

7.1.1 MVC

Since the 1970s the Model View Controller (MVC) pattern is the standard software design pattern for applications that present the user with a graphical user interface. It was developed out of a need for modularity, to encapsulate responsibility of specific concepts to separate software modules. MVC identifies three main components that program code should be grouped into, namely [18]:

> **Model**: The representation of some object of knowledge, encapsulates code managing the associated data and behaviour (business-logic).
>
> **View**: The visual representation of the model. The view can feature or hide aspects of the model and thus act as a presentation filter. The view observes the model for changes and updates the presentation accordingly.
>
> **Controller**: The controller allows the user to interact with the model. It allows the user to trigger behaviours implemented in the Model.

In the classic MVC pattern the model does not ”know” about the view or the controller. And the controller does not effect the view. Instead, both the view and controller monitor the model using an observer mechanism and synchronise themselves when updates to the model occur.

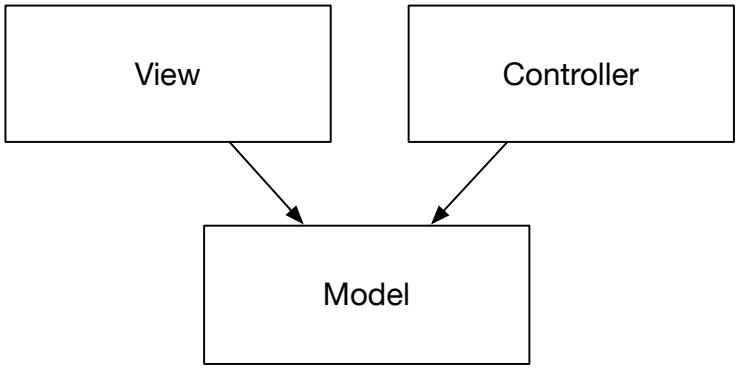

Figure 7.1: Classic MVC

7.1.2 Modern MVC

Modern MVC has evolved from being a software design pattern which handles components of an application to an architectural design pattern that defines the structure of an application itself. It is similar to a Layer Architecture where each layer interacts directly only with the next layer directly above or below [10]. The division of responsibilities is slightly different from the typical presentation,, domain, and data layer definitions.

Most modern web frameworks such as Ruby on Rails, Symfony or the iOS environment refer to themselves as MVC based frameworks. The modern MVC concept has changed slightly. The component definitions are the same, but some responsibilities have shifted. Modern MVC strictly separates the model from the view. All modern frameworks state that the view should have as little logic as possible. Any logic implemented in the view should only be relevant to presentation. The view components should not directly reference the model components [7] [1].

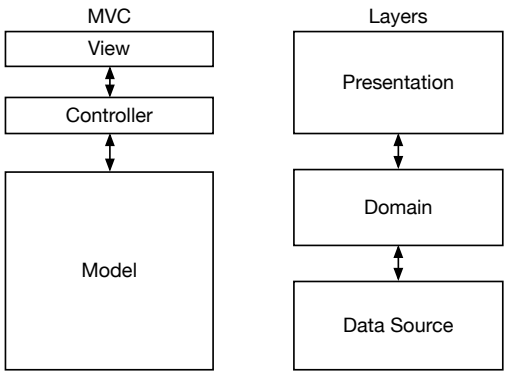

Figure 7.2: Modern MVC vs 3-Layered Architecture

This division of responsibility has the added benefit of increased testability. GUIs are difficult to test, by removing as much logic as possible from the user interface there is less necessity to test it. The controller and model components can more easily be tested in isolation using normal unit tests [11].

7.1.3 MVC Derivatives

Many MVC derivatives exist. The main differences are where the division of responsibility is made and how it is labeled. MVVM defines a view model instead of a controller. The view model acts as a facade around the model and introduces a data binder element that is responsible for keeping the view and view model synchronised. The MVT, calls the view a template and the controller a view. The slight difference is that the template is basically a static file, with no logic, and placeholders for the data. The Django web framework uses MVT terminology, but there is little difference to the other MVC derivatives such as MVP (model view presenter). The AngularJS web framework attempts to end the discussion of which design to follow by labelling itself a MVW (Model View Whatever) framework [14].

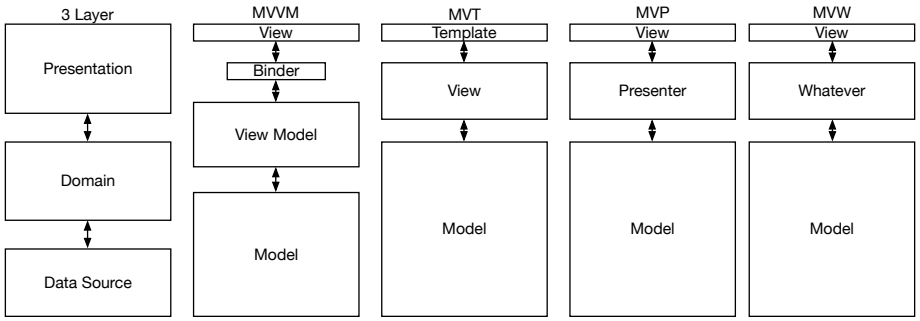

Figure 7.3: MVC Derivatives in relation to 3-Tier Layer

7.2 Mobile Development Strategies

7.2.1 Pure Native vs Hybrid Native vs Hybrid vs Web Apps

The mobile app market is segmented into several platforms. Each platform has it's own framework, API, and language specifications and requirements. Android apps are written in java and access the Android APIs, iOS apps in Objective-C or Swift and interface with the iOS frameworks. Android and iOS cover about 94% of the worldwide mobile market [3]. A developer must decide which platforms are priorities, and how might programming effort be consolidated across multiple platforms. Several strategies exist, and there are many tool kits and frameworks that can help accelerate development.

Web Apps: The simplest development strategy is to develop an HTML based web page that is disguised to appear and behave like a mobile app. The web app runs in a normal browser, is deployed from a standard web server. Model mobile web browsers offer javascript APIs that can access various other hardware components of the smart phone such as the compass, GPS/geolocation services, accelerometer, and the camera. Other services will remain inaccessible.

Hybrid: A similar strategy is to develop using javascript and html, but to package these elements into a "hollow" native app that is little else than a native view component with an embedded web browser. The web browser url is locked to the internal html files and appears to the user as a mobile app. though the user interface might slightly different that a pure native app. The advantage of a hybrid strategy is that the app can be distributed via the standard app stores increasing it's marketability. It can also be extended with native components that might otherwise not be accessible to to a web app such as internal file storage and native data storage mechanisms.

Pure Native: The developer uses the development environment as provided by the phone manufacturer and has full access to all of the native hardware APIs, development kits and tools. The performance and responsiveness of a pure native app is much better compared to web and hybrid solutions. The user interface is built with the standard UI tools allowing a better user experience because the user is presented with familiar UI elements.

Hybrid Native: Recently several cross platform application frameworks have emerged that allow the developer to use a single language that can be compiled across several platforms. These frameworks unify the system and hardware APIs from different platforms into a single API. The limiting factor is how widely the unified API covers all the features the MDApp must integrate.

7.3 Mobile Frameworks

9 Cross Platform Mobile Frameworks were identified and compared. The 9 Frameworks were found using the online Mobile Frameworks Comparison Chart. The target platforms were Android and iOS. The hardware features selected were Camera, Capture, Connection, File and Storage. In addition the framework should include UI Widgets and a free License. The following 9 frameworks were listed:

AppGyver: AppGyver is a hybrid solution based on Cordova/Phonegap. Cordova/Phonegap apps are programmed in javascript, using standard html/css for the interface, and wrapped in an application that is little more than a full screen web browser. Native functionality is made available through plugins. The AppGyver Supersonic device API does not allow a camera preview to be integrated into the UI or the app, this must be implemented using a 3rd party cordova plugin. The AppGyver Supersonic UI can be used to implement the UI Components of the app, or another UI Framework such as ionic could be integrated. In fact

the only difference from AppGyver and ionic is that many AppGyver features utilises an online developer environment which is a commercial service.

Application Craft: Like AppGyver, Application Craft is a based on Cordova/-Phonegap. But the developer environment and deployment process is totally bound to the commercial online service. It is not clear from the developer documentation whether the applications are extendable with 3rd party plugins.

Corona: Cross-platform native solution, targeted toward game development. C++ and Lua are used for programming. The UI features are game oriented and do not offer many advantages for general mobile apps development.

Framework7: This is a professional looking cordova based solution. The UI features are excellent. Native functionality is extendable through 3rd party cordova plugins. Unfortunately Android is only partially supported.

Gideros: Gideros is a cross platform native solution that uses Lua for programming targeted toward game development. The feature set is similar to Corona. So are the limitations. The UI is not appropriate to general app development. The developer documentation is not clear regarding camera integration.

ionic: ionic is a mature UI framework built on Cordova/Phonegap. ionic is implemented using angular.js and offers a rich set of template directives which can be used to customise the user interface easily. Two-way data binding between logic controllers and view templates is also enabled through angular. The ionic developer also provides an online service that assists in development and deployment. Use of the service is completely optional though.

Kivy: Kivy is a python based framework that is compiled to native code. Unfortunately the UI components are not targeted toward general app development and are more game oriented.

The-M-Project: This is a hybrid cross platform framework that uses web technologies to provide the UI and logic. It leverages backbone.js to make UI development simpler. Its not clear in the documentation how camera access is enabled and to what extent. The framework does not appear to be based on Cordova/-Phonegap so those 3rd part plugins might not be available.

ViziApps: Similar to AppGyver and Application Craft. This framework is Cordova/Phonegap based with a strong dependency on a commercial online development and deployment environment.

7.3.1 Comparison

Cross-platform Support: A minimum requirement for cross-platform support is that the framework allows the application to run on iOS and Android devices without any addition configuration.

Enabler of: OE-1, POR-1

Integrated Camera Access: The framework allows integrated access to the devices camera APIs. The MDApp will not simply pass the Smartphone User to the native camera mode. Instead the camera preview will be integrated in the MDApp user interface. If available on the native device the MDApp should also be able to set the focal range, center the focal point and be notified via a callback when the camera lens has focused so that it can notify the Smartphone User that it is ready to capture an image.

Enabler of: Image.Capture.Preview, Image.Capture.Trigger, SI-1.*

Network Connectivity: The framework allows the MDApp to connect via the internet to the online services required to perform the border analysis and risk assessment calculations. This includes uploading and downloading image files as well as posting and receiving data from an online API.

Enabler of: CI-1.*, CI-2.*, Border.Extract.Calculate, Risk.Assessment.Calculate

Native File Access: The framework allows the MDApp to save image files, captured by the camera or downloaded from the border extraction service, to be saved internally on the native file system. Image files shall not only be saved in the devices image library but within the MDApp's allocated storage. On iOS the image files shall be saved in the app internal "/Directory" folder. This directory is accessible and manageable by the user, it is also backed up via the iOS native backup strategies that the Smartphone User would be familiar with. On Android, image files and data should be saved on the shared "External Storage" directory. This ensures that files and data are both accessible, backup-able, and persisted in the case of a software update.

Enabler of: SI-2, INT-1, INT-2, INT-3

Native DB Storage: Hybrid Apps generally live in a browser based context. Persistence of data is an issue therefore because in browsers data is generally transient. The framework chosen needs to offer a data solution either be tying in to native system data solutions (CoreData on iOS for example) or by providing an Sqlite file based solution.

Enabler of: INT-1, INT-2, INT-3, Metadata.Location.*, Metadata.Text.*, Archive.Save, Archive.Browse, Archive.Select.*

UI Components: The framework should include graphical user interface components and layout tools that assist in building a user interface that emulates the native look and feel of the platform. At the same time it should be consistent across all platforms. These are two conflicting requirements so some compromise is necessary to balance these aspects.

Enabler of: UI-5, UI-6, POR-1

Unrestrictive License: The framework should not be prohibitive with respect to intellectual property or financial cost. It should be open source with no restrictions on the distribution model for the app. The financial cost should be low, or free.

Unrestrictive License: Some frameworks are integrated with a service (usually commercial) that allow or restrict development to an online web based development environment. The chosen framework should not restrict development or deployment in any way.

Advanced MVC Integration: Does the framework offer tools that assist the developer to adhere to a modern MVC architecture? Integrated data-binding functionality, for example, allows data synchronisation between the view and controller.

Enabler of: TES-1, MOD-1

Figure 7.4 shows the comparison matrix applied to the 9 mobile cross-platform development framework.

	AppGyver	Application Craft	Corona	Framework7	Gideros	ionic	Kivy	The-M-Project	ViziApps
Cross-platform Support	Y	Y + WindowsPhone + Blackberry	Y	Y- (only partial android and blackberry support)	Y	Y	Y	Y + WindowsPhone + Blackberry	Y
Integrated Camera Access	?	?	?	Y*	?	Y*	(partial feature set)	Y*	?
Network Connectivity	Y*	Y	Y	Y*	Y	Y	Y	Y	Y*
Native File System Access	Y*	Y	Y	Y*	Y	Y*	Y	Y*	Y*
Native DB Storage	Y*	Y	Y	Y*	Y	Y*	experimental, API subject to change	Y*	Y*
UI Components	Y	Y	N	Y	N	Y	Y	Y	Y
Unrestrictive License	Y	Y	Y	Y	Y	Y	Y	Y	?
Unrestrictive Service	N	N	Y	Y	Y	Y	Y	Y	N
Advanced MVC Integration	Y*	N	N	Y	N	Y	N	N	?

* Functionality extendable through plugins

Figure 7.4: Mobile cross platform frameworks comparison

Framework7 and ionic score the best in the comparison chart above. Framework7 is a newer framework with no external dependencies. Ionic has been around longer and is basically a mobile UI framework that is built upon AngularJS, bringing with it all the features that make browser based development easier. Two-way data binding, templating, and RESTful programming resources are some of the highlights of Angular. Because of these features and the widely available programming resources, references, and documentation ionic has been chosen as the framework to implement the MDApp.

7.4 Application Structure

7.4.1 Components

Figure 7.5 shows a high level view of the logical layout of the components in the MDApp.

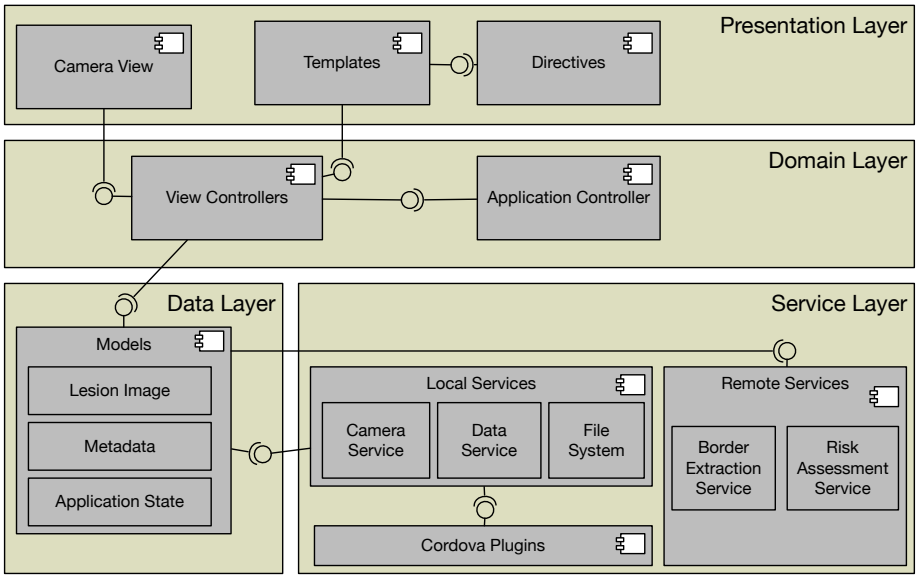

Figure 7.5: Component diagram of the MDApp

7.4.1.1 View Controllers

Each view controller class is responsible for a specific view or partial view of the MDApp. In larger applications there might be multiple view controllers per "page" in a nested hierarchy. The MDApp will only require one controller per page, these are the HomeController, CameraController, AnalysisController and the ArchiveController. Each controller is responsible for preparing the data that will be displayed in the view as well as capturing and processing user events.

7.4.1.2 Application Controller

The application controller is a global controller that can manage the state of the app. It makes sure that the state is persevered when the MDApp is closed and restarted. It can also switch the user automatically from one view to another when appropriate or enable or disable the user interface when some activity is in progress.

7.4.1.3 Templates, Directives and Camera View

Except for the the Camera View all of the pages are created from HTML files that have prepared placeholders called directives. Directives are special markers in the HTML that the Angular framework uses to inject data and specific behaviour into the HTML element. Directives bundle together often used patterns into easily configurable markers that can be embedded into an HTML file.

The Camera View is an exception in the MDApp. The CameraView is an HTML template that overlays a realtime preview of the camera's input. The realtime preview is not part of the Cordova/Phonegap browser view, it is a platform native view element that is provided by a 3rd party plugin. The cordova-plugin-camera-preview is a crossplatofrm wrapper around native code that allows the browser based javascript to communicate and control the camera preview view.

7.4.1.4 Models

The Models are relatively simple data classes. The MDApp does not require much buisiness logic for the data. The Models encapsulate some communication with some of the services so that the controllers can remain as simple as possible.

7.4.1.5 Local Services

The local services classes are just wrappers around the Cordova plugins. Cordova plugins offer a "raw" javascript API. The service classes will hide some of the complexity by making only the relevant functions available to the MDApp.

7.4.1.6 Remote Services

The remote services leverage the angular $resource factory which provides easily configurable RESTful communication to online servers.

7.4.1.7 Cordova Plugins

In a hybrid architecture most of the logic is programmed in javascript, just like a normal web page. Communication with native system components and hardware APIs that a normal browser based environment does not have access can be enabled through plugins. Ideally these plugins unify the native APIs across all platforms. The MDApp uses the following cordova plugins:

7.4.1.7.1 cordova-plugin-camera-preview

Several camera plugins exist for Cordova, however none of them allow a live camera preview to be embedded in the UI directly. Most pass the user to a native camera view. This makes it impossible for any UI components to be superimposed over the camera view. The cordova-plugin-camera-preview makes this possible by creating a native view component that can be positioned over or under the Cordova browser view. In

order to overlay and components in the Cordova browser over the camera, the Cordova browser background must be made transparent. This can be done by setting the background of the html, body and some ionic components transparent.

7.4.1.7.2 cordova-plugin-file-transfer

This plugin allows javascript running in the Cordova browser context to interface with a collection of native classes that manage file downloads and uploads.

7.4.1.7.3 Cordova-sqlite-storage

This plugin creates and manages connectivity to a local sqlite database file that is not subject to the limitations of a browser based file storage. Queries are faster, the storage capacity is much larger, and most importantly data will survive an application restart and update. A backup of the sqlite database file can be made using standard native backup strategies. This is not possible using the built in Cordova browser based data storage

7.4.2 Classes

The class diagram in figure 7.6 shows the structure and connections between the main classes of the MDApp. The service layer classes are singletons that encapsulate the functionality of some plugin into an easy to manage form. Static class level methods and attributes are underlined. The other methods and attributes have object level scope.

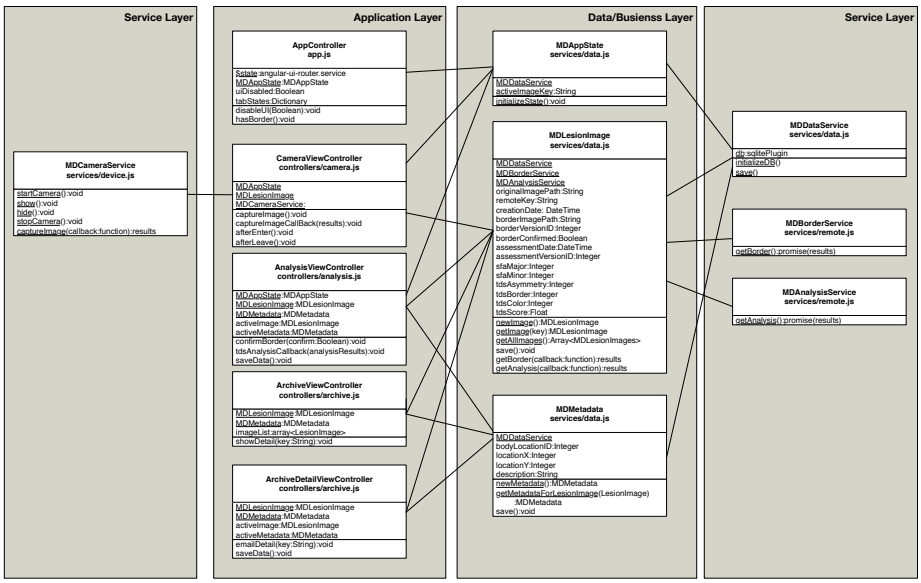

Figure 7.6: Component diagram of the MDApp

7.4.2.1 Application Layer

7.4.2.1.1 AppController

Name	AppController			
Description	The AppController manages the global state of the app. It can disable UI elements when the app needs to wait for longer processes. It can restore the previous state of the app after it has been reopened or updated.			
Type	Class			
Implemented In	app.js			
Attributes	Name	Type	Description	
	$state	class	The $state object is provided by the angular ui-router and is used to define pages in the app. It can be used to query or set the ui state. The AppController uses the $state to automatically move the user from the camera view to the analysis view when the border data has become available, for example.	
	MDAppState	class	The MDAppState stores data the the AppController might need to restore it's state after a restart.	
	uiDisabled	boolean	This variable is used to trigger changes in the ui. Through 2-way binding the ui components in the view will become disabled or enabled respectivly.	
	tabStates	object	This javascript object stored the disabled-state of the individual tab components in the ui. The AppController disabiles or enables specific tab elements based on the state of the app and the data currently available.	
Methods	Name	Parameter	Return Type	Description
	diableUI	shouldDisable : boolean	void	sets or unsets the state of uiDisabled and tabStates according to the shouldDisable parameter.
	hasBorder	void	void	moves the user to the AnalysisView when a border preview image has become available for confirmation.

Table 7.1: AppController Specification

7.4.2.1.2 CameraViewController

Name	CameraViewController		
Description	The CameraViewController manages the camera preview user interface. It received a user "tap" event from the view and triggers the image capture process. Because the actual camera view ui element is a native view element outside of and behind the Cordova browser view, specific ui components must be made transparent to let the camera view become visible. The CameraViewController activates this transparency when it is loaded and deactivates it when unloaded.		
Type	Class		
Implemented In	controllers/camera.js		
Attributes	Name	Type	Description

	MDCameraService	class	The CameraViewController passes the capture request from the user to the MDCameraService. The MDCameraService is also responsible for starting and stopping the camera preview.	
	MDLesionImage	class	When a new image is captured the CameraViewController notifies the MDLesionImage class which creates a new MDLesionImage instance.	
	MDAppState	class	When a new image is captured, the CameraViewController notifies the MDAppState that a new image is the active image.	
Methods	Name	Parameter	Return Type	Description
	captureImage	void	void	This method is triggered by the user and passed from the view to the view controller. The event is passed on to the MDCameraService along with a reference to the captureImageCallback which is called when an image has been saved.
	captureImageCallback	results	void	The MDCameraService notifies the CameraViewController that an image is availble via this callback. The results parameter is a javascript object that contains a path to the captured image file and a preview image file.
	afterEnter	void	void	When the camera view is loaded this method is called by the angular ui-router. The CameraViewController maked selected background UI components transparent in order to allow the camera preivew view to become visible.
	afterLeave	void	void	The angular ui-router notifies the CameraViewController via this method that the user has moved to another view. The CameraViewController cleans up the transparent UI compontent, making them opaque.

Table 7.2: CameraViewController Specification

7.4.2.1.3 AnalysisViewController

Name	AnalysisViewController		
Description	The AnalysisViewController manages the analysis view and provides the functions that let the user confirm the border calculation and edit metadata associated with a captured image.		
Type	Class		
Implemented In	controllers/analysis.js		
Attributes	Name	Type	Description
	MDAppState	class	The AnalysisViewController gets the id of the currently active image from the MDAppState service.

	MDLesionImage	class	The AnalysisViewController gets the instance of the currently active image from the MDLesion-Image service.	
	MDMetadata	class	The AnalysisViewController gets the instance of the MDMetadata associated with the currently active image.	
	activeImage	instance of MDLesion-Image	The MDLesionImage instance currently presented to the user.	
	activeMetadata	instance of MDMetadata	The MDMetadata instance currently presented to the user.	
Methods	Name	Parameter	Return Type	Description
	confirmBorder	confirm : boolean	void	This method is triggered by the user and passed from the view to the view controller. The event is passed on to the MDImageService along with a reference to the tdsAnalysis-Callback function which is called when an tds analysis has completed.
	tdsAnalysisCallback	results	void	The MDImageService notifies the AnalysisViewController that the analysis is complete via this callback. The results parameter is a javascript object that contains the tds analysis information.
	saveData	void	void	This method saves the results of the tds analysis, border information and any metadata associated with the lesion image.

Table 7.3: AnalysisViewController Specification

7.4.2.1.4 ArchiveViewController

Name	ArchiveViewController			
Description	Controller for the archive list view. This Controller reads data from the MDLesion-Image and MDMetadata services which is presented to the user as a list in the archive view. The user can select an item from the list and will be transitioned to the archive detail view. The selection and transition logic is managed by the angular ui-router service directly in the view.			
Type	Class			
Implemented In	controllers/archive.js			
Attributes	Name	Type	Description	
	MDLesionImage	class	The MDLesionImage service provides a list of previously captured images and results which can be reviewed.	
	MDMetadata	class	THe MDMetadata service provides associated metadata to be attached to the image list in the view.	
Methods	Name	Parameter	Return Type	Description

Table 7.4: ArchiveViewController Specification

7.4.2.1.5 ArchiveDetailViewController

Name	ArchiveDetailViewController			
Description	Controller for the archive detail view. The id of the selected LesionImage is passed to this controller when loaded. The controller prepares the data of the selected archived image to be presented to the use in the archive detail view. This controllers methods are responsible for saving edited metadata back to the archive or sending images and data via email.			
Type	Class			
Implemented In	controllers/archive.js			
Attributes	Name	Type	Description	
	MDLesionImage	class	MDLesionImage provides the controller with an instance of an MDLesionImage.	
	MDMetadata	class	MDMetadata provides the controller with metadata associated with an instance of MDLesionImage	
	activeImage	instance of MDLesionImage	This is the selected instance of MDLesionImage to be presented to the user in the detail view.	
	activeMetadata	instance of MDMetadata	This is an instance of the metadata assciated with the selected MDLesionImage instance.	
Methods	Name	Parameter	Return Type	Description
	emailDetail	key	void	the data from the activeImage and activeMetadata is flattended to an html template and passed to the email service.
	saveData	void	void	Any user edits to the metadata will be saved.

Table 7.5: ArchiveDetailViewController Specification

7.4.2.2 Service Layer

7.4.2.2.1 MDDataService

Name	MDDataService			
Description	The MDDataService is a singelton class that encapsulates all the database access and management functions. MDDataService wraps two external services, the Cordova-sqlite-storage plugins and the persistence.js javascript library. The Cordova-sqlite-storage plugin manages connectivity to the sqlite database file. Persistence.js is an ORM library that hides away any messy sequel development behind a nice object oriented api.			
Type	Class			
Implemented In	services/data.js			
Attributes	Name	Type	Description	
	db	object	The database object provided by the Cordova-sqlite-storage plugin.	
Methods	Name	Parameter	Return Type	Description
	initializeDB	void	void	This opens a connection to the sqlite database file and creates the table structure in the database if it does not already exist. This method must be called after the Cordova environment has loaded and the application context is ready.
	saveData	void	void	This method saves any changes made to instantiated data objects.

Table 7.6: MDDataService Specification

7.4.2.2.2 MDBorderService

Name	MDBorderService			
Description	The MDBorderService is a singelton class that encapsulates communiation with the remote border extraction service. The MDBorderService encapsulates communcation to the cordova-plugin-file-transfer plugin. Results from the plugin are wrapped into an angular defered promise allowing other angular components to more easily integrate the file transfer services.			
Type	Class			
Implemented In	services/remote.js			
Attributes	Name	Type	Description	
Methods	Name	Parameter	Return Type	Description
	getBorder	void	promise (results)	This function prepares the file upload configuration parameters and initiates the upload. The success and error callbacks are wrapped in an angular $q promise that is retured to the caller and can be evaluated in a local anonymous function when it resolves.

Table 7.7: MDBorderService Specification

7.4.2.2.3 MDAnalysisService

Name	MDAnalysisService			
Description	The MDAnalysisService is a singelton class that encapsulates communiation with the remote image analysis service. Analysis of a previoulsy upload image may be triggered via an online rest API. Results from the analysis are wrapped into an angular defered promise.			
Type	Class			
Implemented In	services/remote.js			
Attributes	Name	Type	Description	
Methods	Name	Parameter	Return Type	Description
	getAnalysis	imageKey	promise (results)	This function sends a request to the online analysis service via a rest api and wraps the response into an angular promise that can be resolved by the calling class in a local function asynchronously.

Table 7.8: MDAnalysisService Specification

7.4.2.2.4 MDCameraService

Name	MDCameraService
Description	This Service encapsulats communication to the cordova-plugin-camera-preview plugin. It wraps asyncronous methods and callbacks in angular.js promises that allow other classes to integrate this service in an angular-like fashion.

Type	Class			
Implemented In	services/device.js			
Attributes	Name	Type	Description	
Methods	Name	Parameter	Return Type	Description
	startCamera	void	void	This method starts the camera and activates the camera preview view.
	captureImage	void	promise (results)	Initiates the capture method in the cordova plugin and passes a promise to the caller that can be resolved locally.
	stopCamera	void	promise (results)	Hides the camera preview view and stops the camera.

Table 7.9: MDCameraService Specification

7.4.2.3 Data Layer

7.4.2.3.1 MDAppState

Name	MDAppState			
Description	This service manages data that describes the current state of the app. If the app is restarted or updated the MDAppState service will persist the state data so it can be reloaded when the app is restarted.			
Type	Class			
Implemented In	services/data.js			
Attributes	Name	Type	Description	
	MDDataService	class	The MDAppState communicated with the MDDataService in order to query the sqlite database file.	
	activeImageKey	String	If an image had been captured before the app was perviously shutdown this attribute will hold the corresponding image key which can be used to instantiate an MDLesionImage and associated Metadata.	
Methods	Name	Parameter	Return Type	Description
	initializeState()	void	void	This method checks if application state data is available, if so it populates the corresponding attributes. It also initializes any state data if the appliation is started for the first time.

Table 7.10: MDAppState Specification

7.4.2.3.2 MDLesionImage

Name	MDLesionImage		
Description	The MDLesionImage service		
Type	Class		
Implemented In	services/remote.js		
Attributes	Name	Type	Description
	MDDataService	class	The service that encapsulates the connections to the database.

	MDBorderService	class	Encapsulates the connection to the remote border calculation service	
	MDAnalysisService	class	Encapsulates the connection to the remote image analysis service	
	originalImagePath	String	path to the original image file.	
	remoteKey	String	Uuid used to match the local image to data stored on the online service. Generated by the server.	
	creationDate	DateTime	The date when the original image was captured.	
	borderImagePath	String	Path to the image file that contains the border data	
	borderVersionID	Integer	Version number of the algorithm used to calculate the border.	
	borderConfirmed	Boolean	This value is set to true when the user has confirmed that the border has been calculated precisely.	
Methods	Name	Parameter	Return Type	Description
	newImage	void	MDLesion-Image	Creates and returns a new instance of a MDLesionImage
	getImage	key : String	MDLesion-Image	Returns an instance of MDLesion-Image corresponding to the key value paramter.
	getAllImages	void	array-<MDLesion-Image>	Returns an array of all stored MDLesionImage instances.
	save	void	void	Saves changes to all modified MDLesionImage instances.
	getBorder	image : MDLesion-Image, callback : function	results	Calls the MDBorderService to initiate calculation of the border. Returns the results to the caller via the callback function.
	getAnalysis	image : MDLesion-Image, callback : function	results	Calls the MDAnalysisService to initiate calculation of the analysis. Returns the results to the caller via the callback function.

Table 7.11: MDLesionImage Specification

7.4.2.3.3 MDMetadata

Name	ArchiveDetailViewController		
Description	Controller for the archive detail view. The id of the selected LesionImage is passed to this controller when loaded. The controller prepares the data of the selected archived image to be presented to the use in the archive detail view. This controllers methods are responsible for saving edited metadata back to the archive or sending images and data via email.		
Type	Class		
Implemented In	services/data.js		
Attributes	Name	Type	Description
	MDLesionImage	class	MDLesionImage provides the controller with an instance of an MDLesionImage.
	MDMetadata	class	MDMetadata provides the controller with metadata associated with an instance of MDLesion-Image

	activeImage	instance of MDLesion-Image	This is the selected instance of MDLesionImage to be presented to the user in the detail view.	
	activeMetadata	instance of MDMetadata	This is an instance of the metadata assciated with the selected MDLesionImage instance.	
Methods	Name	Parameter	Return Type	Description
	emailDetail	key	void	the data from the activeImage and activeMetadata is flattended to an html template and passed to the email service.
	saveData	void	void	Any user edits to the metadata will be saved.

Table 7.12: ArchiveDetailViewController Specification

7.4.2.4 Partial Sequence Diagram

Figure 7.7 shows show the sequence of events responsible for capturing an image and initiating the border calculation. This covers all of use case UC-1 and the first step of the flow in UC-2. The steps corresponding to the numbers in the figure are descibed below.

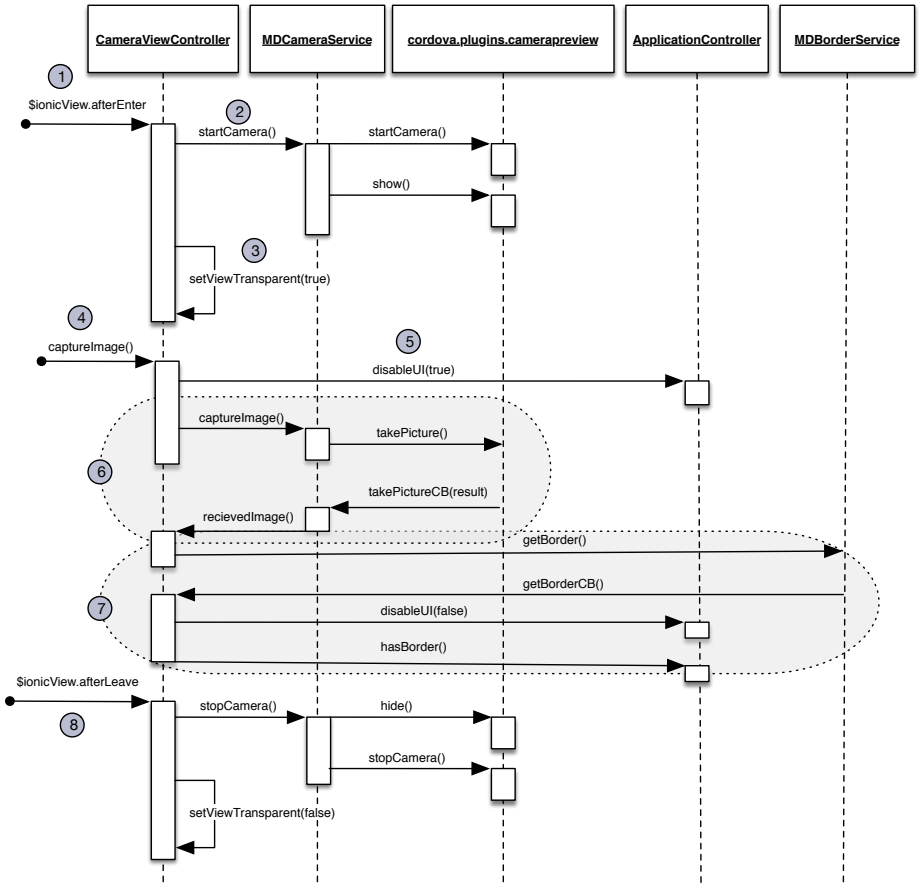

Figure 7.7: Partial Sequence Diagram

1: By clicking on a a menu element or a tab element the user signals to the angular ui-router service that the camera view should be activated. The ui-router loads the CameraViewController and the corresponding ui elements, then send the $ionicView.afterEnter signal to the CameraViewController.

2: The CameraViewController calls the startCamera method on the MDCameraService class. This class in turn takes calls the necessary methods on the plugin that managed the camera preview and ui view element.

3: The CameraViewController selects specific application ui elements and makes their background colour transparent in order to let the camera view element become visible in the background. The user can now position the camera while observing the preview until the optimal position is found.

4: By tapping a finger on the camera preview view an event is created in the view

that calls the captureImage method in the CameraViewController.

5: Before initiating the image capture process, the ui is disabled to that no other event will interrupt the following processes. This is important because another accidental tap on the display could trigger another image capture process before the first one has been reviewed.

6: The image capture process can take some time and is implemented by the plugin as an asynchronous function that notifies the caller via a callback. The MDCameraService class wraps the function and callback in an angular.js process that the CameraViewController can resolve locally in a typical angular manner. Figure 7.8 is an example of the MDCameraService captureImage() implementation. Figure 7.9 illustrates how it can be called and resolved locally in the CameraViewController.

```
.service('MDCameraService', function ($q) {

  this.captureImage = function () {

    var defered = $q.defer();

    cordova.plugins.camerapreview.setOnPictureTakenHandler(function(result){
      defered.resolve(result);
    });

    cordova.plugins.camerapreview.takePicture();

    return defered.promise;

  };


  ..other methods..

})
```

Figure 7.8: MDCameraService captureImage() implementation

```
$scope.captureImage = function () {
  disableUI(true)
  MDCameraService.captureImage().then(function (results) {

    var imagePath = result[0];
    $rootScope.activeImage = MDLesionImage.newImage(imagePath);
    MDAppState.ActiveImageKey = $rootScope.activeImage.id

    MDBorderService.getBorder(result[0]).then(function (response) {

      $rootScope.activeImage.originalImagePath = response.original;
      $rootScope.activeImage.borderImagePath = response.border;
      $rootScope.activeImage.save();

      disableUI(false);

      $scope.$emit('disableUI', false);
      $scope.$emit('hasBorder');

    });
  });
};
```

Figure 7.9: CameraViewController captureImage() implementation

7: The implementation of these interactions can also be seen in figure 7.9. The MDBorderService.getBorder() function is resolved in the returned promise's then() method. The CameraViewController gets the data from MDBorderService and attaches it to the $rootScope element which is accessible globally. The changes to the activeImage element are saved to the database. Then the Controller cleans up the UI and signals to the AppController that the border data is available for review.

8: The ApplicationController calls the ui-router (not shown in figure 7.7) that the user should be transferred to the Analysis View. This unloads the camera view and triggers the $ionicView.afterLeave method in the CameraViewController. The CameraViewController can now safely close the camera preview view and clean up the ui before unloading.

7.5 User Interface

This section describes the components of the user interface. The components are grouped into views which allow the user to interact with a particular state or feature of the application. Some views may extend beyond the edges of the smart phone's screen size, in that case the figure will attempt to show multiple versions of a view. The figures shown do not represent every possible combination of states the application may have.

7.5.0.0.1 Home Screen

The Home Screen is the view that will appear to the user when the MDApp is first started. It will act as a jump off point to guide the user through the app, showing quick links with a short description. When the MDApp is started for the first time, there will be no data or images available to view in the Analysis or Archive views. The MDApp will therefore not display the quick link for the Analysis View or the Archive View, only the Camera View.

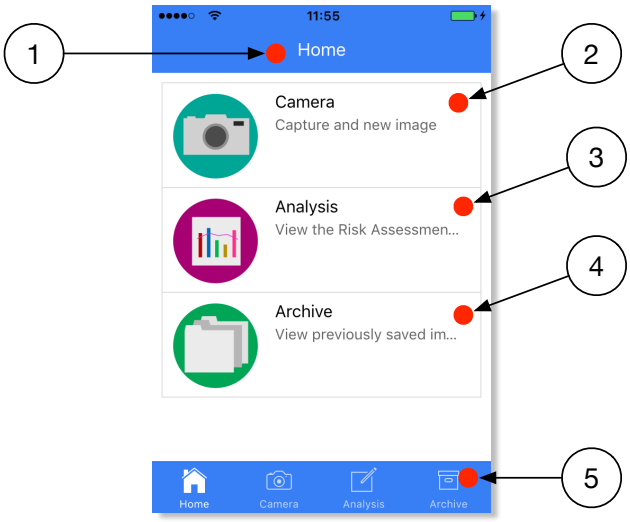

Figure 7.10: Home Screen

1 View Title, every view shall have a title.

2 Quick link to the Camera View.

3 Quick link to the Analysis View, only visible if an image has already been captured.

4 Quick link to the Archive View, only visible if images have been saved to the archive.

5 Tab-bar navigation element. This component is always visible, in all of the app's views.

7.5.0.0.2 Camera View

The Camera View will automatically start the real time camera preview, displaying an image of what the camera is currently pointing at. Some visual indicators will help the user to center the skin lesion in the image. Because the user might be taking an image of her or his own body it might be difficult to click a button while holding the camera

steady. Therefore touching anywhere on the camera preview area will trigger an image capture. An audible "shutter" noise will also signal the user that an image has been captured.

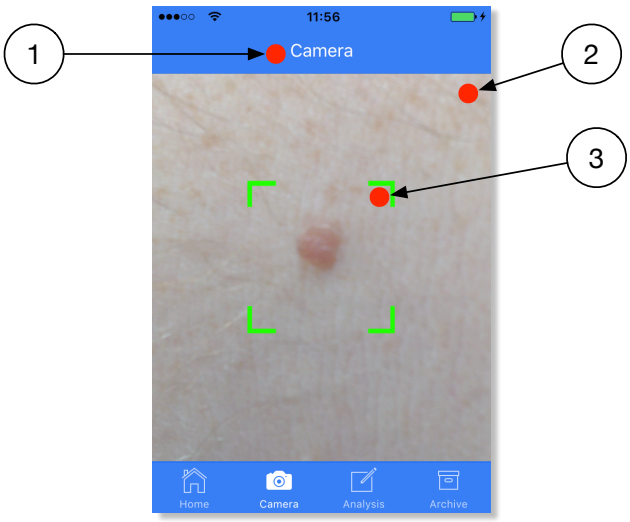

Figure 7.11: Camera View

1 View Title, every view will have a title.

2 Realtime camera preview.

3 Crosshairs, visual markers that indicate where the center of the image is. The user should attempt to keep the lesion within the markers, but filling the marked area as much as possible. (limited by the focal range of the smart phone's camera)

7.5.0.0.3 Analysis View

The Analysis View presents the current state of processing and the results of any finished calculations. When the border calculation is finished, the Smartphone User is promoted to confirm that the border was precisely calculated. If not the app will return the Smartphone User to the Camera View to capture another version of the image. The risk assessment data will be presented to the Smartphone User below the border image along with a summary and recommendation as text (not shown in figure ref[]). The Smartphone User can also add addition text as metadata and specify where on her/his body the skin lesion was located. The Smartphone User also has the option to save the image and data to the archive.

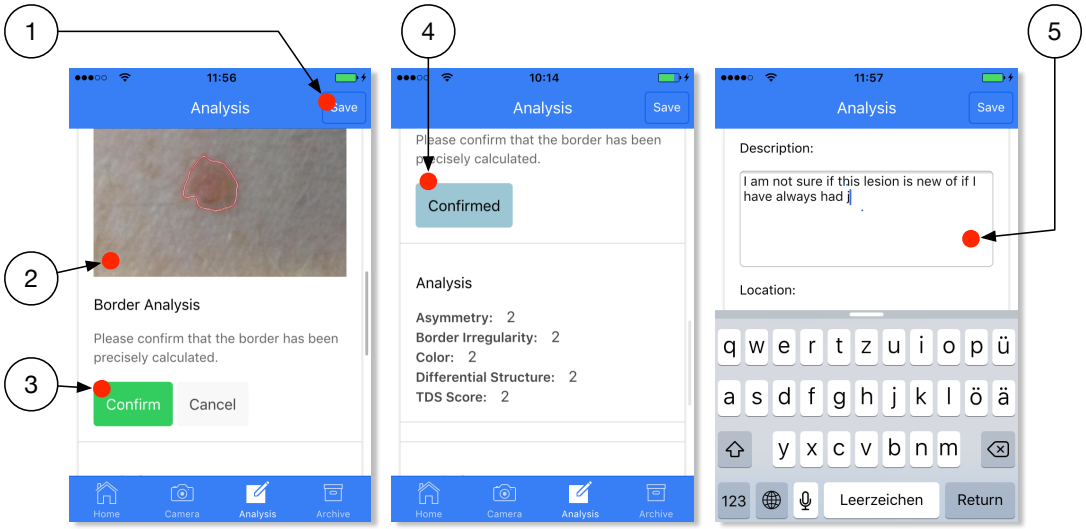

Figure 7.12: Analysis View

1 Save button.

2 Border Analysis preview

3 Confirm and Cancel buttons with which the Smartphone User may confirm or reject that the border has been precisely calculated.

4 Once confirmed, the button becomes an indicator that the border was confirmed. It is no longer clickable in this state.

5 Additional metadata may be entered by the Smartphone User as text. Upon clicking a text input field the smartphone's native text entry keyboard element will appear.

7.5.0.0.4 Archive List View

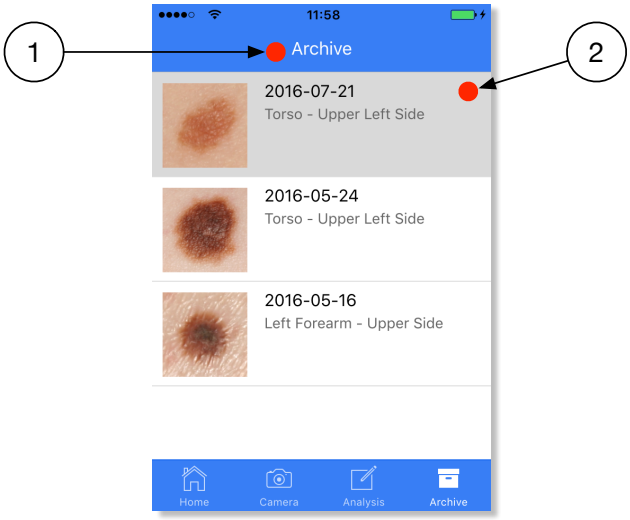

Figure 7.13: Archive List View

1 View Title.

2 Clickable list of saved lesion images and associated data. When clicked the Archive Detail View will load and present the corresponding data.

7.5.0.0.5 Archive Detail View

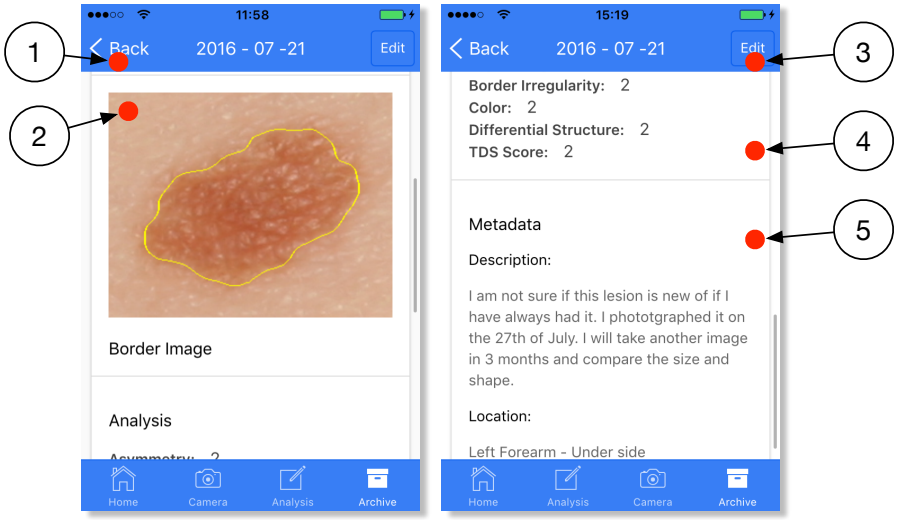

Figure 7.14: Archive Detail View

1 Back button will transfer the SmartphoneUser back to the Archive List View.

2 Border Analysis preview

3 Edit button will make the saved Metadata editable. A save and cancel button will allow the Smartphone User to save changed made or revert to the previous state before editing.

4 TDA analysis results.

4 Metadata associated with the Lesion Image.

Chapter 8

Conclusions

The goals of this project have mostly been completed. Some topics were more complex than originally estimated and some compromises were made regarding the details of each topic. This project can be regarded as a high level overview of the components necessary to build a smart phone based melanoma detection application. Several topics could probably become the basis of further dedicated projects and expanded on. Each goal is reviewed in more detail below.

1 The current state of the available medical mobile apps was explored. The apps were grouped into categories and compared. While the overall majority of apps were providing an educational service, many diagnostic apps were identified. In comparison with other research from 2013 it could be shown that the availability of apps that provide a diagnostic function is increasing.

2 A large part of the effort that went into this project was dedicated to researching and becoming familiar with image processing and analysis algorithms. During the exploration and experimentation of methods to automatically identify the skin lesion some novel insights were gained.

3 Much research exists regarding automatic risk assessment of melanoma in images captured with a dermatoscope, less so for images captured with a smart phone camera. The difficulty this introduces became apparent. It remains unclear after this project how much the methods for dermatoscopic image risk assessment are actually applicable to images captured with a smart phone.

4 One of the stated goals was to choose a specific algorithm and data with which to proceed. The choice of algorithm became limited by the time available for research and prototyping. Basically what worked in the experimentation phase was chosen. The time it would have taken to implement and compare several algorithms was not available.

5 Putting the research together into a working algorithm was not difficult as far as programming was concerned. The python based prototypes form the experimentation phase could be reused easily in the final algorithm. The most effort was

spent in actually reviewing results and trying to achieve and overview of how well the different components actually worked on all the sample image sets. A lot of development time went into building tools that accelerated the review and comparison effort. Even with the tools though it was not possible to do much comparing and optimisation because the review process still took several days to complete. So, although an algorithm was implemented that could identify and assess a skin lesion image, it's performance has not been adequately reviewed and optimised.

6 A concept was created for a mobile medical app that would provide the user with a risk assessment based on the algorithms implemented. The amount of effort required for this was initially underestimated due to an unfamiliarity with requirements engineering principals. After several iterations and time spent learning the concepts a relatively complete concept was created. It became clear that requirements engineering is not a static process but requires several iterations where insights gained in later steps must flow back into the requirements definitions. This will be a valuable takeaway from this project.

7 A proof of concept app was developed in tandem with the requirements engineering process. The proof of concept evolved from a static mockup into an actual app and was used to evaluated and refine the requirement definitions. Although the proof of concept itself is not described in this document, the concepts described in the requirements and design chapters flow from the evolution of the prototype.

Bibliography

[1] Symfony Book: Chapter 2 - Exploring Symfony's Code, The MVC Pattern, 2015. [Online; accessed 2016-8-27], link.

[2] Image noise - Wikipedia, 2016. [Online; accessed 2016-9-23], link.

[3] Mobile/Tablet Operating System Market Share, 2016. [Online; accessed 2016-8-28], link.

[4] Skin Cancer Overview — Patient Version, 2016. [Online; accessed 2016-9-2], link.

[5] SLIC Superpixels, 2016. [Online; accessed 2016-9-23], link.

[6] Jose Fernandez Alcon, Calina Ciuhu, Warner ten Kate, Adrienne Heinrich, Natallia Uzunbajakava, Gertruud Krekels, Denny Siem, and Gerard de Haan. Automatic Imaging System With Decision Support for Inspection of Pigmented Skin Lesions and Melanoma Diagnosis. *IEEE*, 3(1):14–25, feb 2009.

[7] Apple. Model-View-Controller, 2015. [Online; accessed 2016-8-27], link.

[8] Ann Chang Brewer, Dawnielle C. Endly, Jill Henley, Mahsa Amir, Blake P. Sampson, Jacqueline F. Moreau, and Robert P. Dellavalle. Mobile Applications in Dermatology. *JAMA Dermatol*, 149(11):1300, nov 2013.

[9] Mercedes Filho, Zhen Ma, and João Manuel R. S. Tavares. A review of the quantification and classification of pigmented skin lesions: From dedicated to hand-held devices. *Journal of Medical Systems*, 39(11):1–12, 2015.

[10] Martin Fowler. *Patterns of enterprise application architecture*. Addison-Wesley Professional, 2002.

[11] Martin Fowler. GUI Architectures, Humble View, 2006. [Online; accessed 2016-8-28], link.

[12] Jeffrey Glaister. Automatic segmentation of skin lesions from dermatological photographs. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2013.

[13] A.A. Marghoob, J. Malvehy, R.P. Braun, and A.W. Kopf. *An Atlas of Dermoscopy*. Encyclopedia of Visual Medicine. CRC Press, 2004.

[14] Igor Minar. MVC vs MVVM vs MVP, 2012. [Online; accessed 2016-8-28], link.

[15] J. Premaladha and K.S. Ravichandr. Asymmetry Analysis of Malignant Melanoma Using Image Processing: A Survey. *J. of Artificial Intelligence*, 7(2):45–53, feb 2014.

[16] Sophia.M Mallikarjun Mundas Vidya.R Siddiq Iqbal, Divyashree.J.A. Implementation of Stolz's Algorithm for Melanoma Detection. *International Advanced Research Journal in Science, Engineering and Technology IARJSET*, 2(6), jun 2015.

[17] Aurora Sáez, Begoña Acha, and Carmen Serrano. Pattern Analysis in Dermoscopic Images. In *Series in BioEngineering*, pages 23–48. Springer Science & Business Media, sep 2013.

[18] Stephen Walther. The Evolution of MVC, 2016. [Online; accessed 2016-8-27], link.

[19] Ulrike Weigert, Walter H.C. Burgdorf, and Wilhelm Stolz. ABCD Rule. In *Atlas of Dermoscopy*, page 116. Informa Healthcare, aug 2012.

[20] Karl Wiegers. *Software requirements*. Microsoft, Redmond, Washington, 2013.

Chapter 9

Appendix

9.1 TDS Evaluation

Name	Source	Category	Asymmetry	Border	Color	TDS
B1052.png	Dermofit	Malignant Melanoma	2	4	3.0	4.50
B287.png	Dermofit	Malignant Melanoma	2	5	2.0	4.10
B302.png	Dermofit	Malignant Melanoma	0	3	2.0	1.30
B314.png	Dermofit	Malignant Melanoma	2	6	2.0	4.20
B65.png	Dermofit	Malignant Melanoma	2	4	3.0	4.50
C158.png	Dermofit	Malignant Melanoma	0	2	3.0	1.70
C201.png	Dermofit	Malignant Melanoma	2	4	3.0	4.50
C263a.png	Dermofit	Malignant Melanoma	2	4	3.0	4.50
C311b.png	Dermofit	Malignant Melanoma	2	8	3.0	4.90
C359.png	Dermofit	Malignant Melanoma	1	3	3.0	3.10
D143.png	Dermofit	Malignant Melanoma	0	1	3.0	1.60
D39.png	Dermofit	Malignant Melanoma	2	3	3.0	4.40
D630.png	Dermofit	Malignant Melanoma	0	2	1.0	0.70
D678.png	Dermofit	Malignant Melanoma	0	0	3.0	1.50
T233a.png	Dermofit	Malignant Melanoma	1	8	2.0	3.10
T86b.png	Dermofit	Malignant Melanoma	1	3	3.0	3.10
A121a.png	Dermofit	Melanocytic Nevus	0	3	3.0	1.80
A21b.png	Dermofit	Melanocytic Nevus	0	2	2.0	1.20
A2b.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
A8c.png	Dermofit	Melanocytic Nevus	2	4	2.0	4.00
A8d.png	Dermofit	Melanocytic Nevus	1	2	1.0	2.00
A8e.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
B125c.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
B157.png	Dermofit	Melanocytic Nevus	1	2	3.0	3.00
B17a.png	Dermofit	Melanocytic Nevus	1	2	3.0	3.00
B17b.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
B17c.png	Dermofit	Melanocytic Nevus	1	2	1.0	2.00

B17f.png	Dermofit	Melanocytic Nevus	1	3	2.0	2.60
B197d.png	Dermofit	Melanocytic Nevus	1	0	1.0	1.80
B202b.png	Dermofit	Melanocytic Nevus	1	0	3.0	2.80
B293b.png	Dermofit	Melanocytic Nevus	0	1	1.0	0.60
B293d.png	Dermofit	Melanocytic Nevus	0	1	1.0	0.60
B293e.png	Dermofit	Melanocytic Nevus	0	2	2.0	1.20
B311c.png	Dermofit	Melanocytic Nevus	1	4	1.0	2.20
B350a.png	Dermofit	Melanocytic Nevus	0	0	1.0	0.50
B355b.png	Dermofit	Melanocytic Nevus	2	0	1.0	3.10
B356.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
B361a.png	Dermofit	Melanocytic Nevus	0	1	1.0	0.60
B379a.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
B379b.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
B447a.png	Dermofit	Melanocytic Nevus	2	8	2.0	4.40
B447b.png	Dermofit	Melanocytic Nevus	1	3	2.0	2.60
B472b.png	Dermofit	Melanocytic Nevus	0	2	1.0	0.70
B508.png	Dermofit	Melanocytic Nevus	2	5	2.0	4.10
B522b.png	Dermofit	Melanocytic Nevus	0	0	1.0	0.50
B52a.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
B52c.png	Dermofit	Melanocytic Nevus	0	4	2.0	1.40
B543.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
B549c.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
B574.png	Dermofit	Melanocytic Nevus	1	0	2.0	2.30
B598a.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
B612b.png	Dermofit	Melanocytic Nevus	1	2	1.0	2.00
B654c.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
B66c.png	Dermofit	Melanocytic Nevus	0	3	2.0	1.30
B676a.png	Dermofit	Melanocytic Nevus	0	3	2.0	1.30
B676b.png	Dermofit	Melanocytic Nevus	1	4	3.0	3.20
B69a.png	Dermofit	Melanocytic Nevus	1	2	3.0	3.00
B69b.png	Dermofit	Melanocytic Nevus	2	2	2.0	3.80
B89e.png	Dermofit	Melanocytic Nevus	2	1	1.0	3.20
B91a.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
B91b.png	Dermofit	Melanocytic Nevus	1	6	3.0	3.40
B91c.png	Dermofit	Melanocytic Nevus	2	0	2.0	3.60
D144.png	Dermofit	Melanocytic Nevus	2	3	3.0	4.40
D176b.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
D239a.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
D239b.png	Dermofit	Melanocytic Nevus	0	0	1.0	0.50
D271a.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
D291b.png	Dermofit	Melanocytic Nevus	0	2	2.0	1.20
D339.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
D374.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
D384.png	Dermofit	Melanocytic Nevus	0	7	2.0	1.70
D395.png	Dermofit	Melanocytic Nevus	0	0	3.0	1.50
D404.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00

D426.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
D427b.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
D492.png	Dermofit	Melanocytic Nevus	2	1	2.0	3.70
D526b.png	Dermofit	Melanocytic Nevus	0	2	2.0	1.20
D567b.png	Dermofit	Melanocytic Nevus	2	2	1.0	3.30
D626.png	Dermofit	Melanocytic Nevus	0	2	3.0	1.70
D715.png	Dermofit	Melanocytic Nevus	0	0	1.0	0.50
D722.png	Dermofit	Melanocytic Nevus	2	6	2.0	4.20
D723a.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
D726d.png	Dermofit	Melanocytic Nevus	0	7	2.0	1.70
D726e.png	Dermofit	Melanocytic Nevus	1	3	2.0	2.60
P103a.png	Dermofit	Melanocytic Nevus	0	3	2.0	1.30
P126b.png	Dermofit	Melanocytic Nevus	2	1	1.0	3.20
P144.png	Dermofit	Melanocytic Nevus	0	1	1.0	0.60
P18.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P196.png	Dermofit	Melanocytic Nevus	2	3	2.0	3.90
P199.png	Dermofit	Melanocytic Nevus	1	2	1.0	2.00
P2.png	Dermofit	Melanocytic Nevus	0	3	1.0	0.80
P237b.png	Dermofit	Melanocytic Nevus	1	6	2.0	2.90
P256a.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
P271f.png	Dermofit	Melanocytic Nevus	2	2	2.0	3.80
P277b.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P291.png	Dermofit	Melanocytic Nevus	0	0	3.0	1.50
P304a.png	Dermofit	Melanocytic Nevus	1	0	2.0	2.30
P306a.png	Dermofit	Melanocytic Nevus	1	0	3.0	2.80
P306c.png	Dermofit	Melanocytic Nevus	0	0	3.0	1.50
P337a.png	Dermofit	Melanocytic Nevus	0	0	3.0	1.50
P337b.png	Dermofit	Melanocytic Nevus	0	1	3.0	1.60
P337d.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
P337e.png	Dermofit	Melanocytic Nevus	1	1	1.0	1.90
P354a.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
P359b.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
P365d.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
P365e.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P376b.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P376d.png	Dermofit	Melanocytic Nevus	2	5	1.0	3.60
P382a.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
P384b.png	Dermofit	Melanocytic Nevus	1	2	2.0	2.50
P384c.png	Dermofit	Melanocytic Nevus	1	7	2.0	3.00
P392.png	Dermofit	Melanocytic Nevus	0	1	3.0	1.60
P399.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P404a.png	Dermofit	Melanocytic Nevus	1	0	2.0	2.30
P404c.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00
P407b.png	Dermofit	Melanocytic Nevus	0	2	2.0	1.20
P407c.png	Dermofit	Melanocytic Nevus	0	1	2.0	1.10
P432.png	Dermofit	Melanocytic Nevus	2	4	2.0	4.00

P435b.png	Dermofit	Melanocytic Nevus	0	3	3.0	1.80
P454a.png	Dermofit	Melanocytic Nevus	1	1	3.0	2.90
P454b.png	Dermofit	Melanocytic Nevus	0	0	1.0	0.50
P45a.png	Dermofit	Melanocytic Nevus	2	1	3.0	4.20
P45b.png	Dermofit	Melanocytic Nevus	1	5	2.0	2.80
P49.png	Dermofit	Melanocytic Nevus	0	0	3.0	1.50
P505c.png	Dermofit	Melanocytic Nevus	2	1	2.0	3.70
P505e.png	Dermofit	Melanocytic Nevus	1	7	2.0	3.00
P509c.png	Dermofit	Melanocytic Nevus	1	1	2.0	2.40
P53b.png	Dermofit	Melanocytic Nevus	0	6	1.0	1.10
P56b.png	Dermofit	Melanocytic Nevus	0	2	3.0	1.70
P58.png	Dermofit	Melanocytic Nevus	0	3	2.0	1.30
P63.png	Dermofit	Melanocytic Nevus	1	6	1.0	2.40
P88b.png	Dermofit	Melanocytic Nevus	0	0	2.0	1.00

Table 9.1: Results of TDS calculation for Dermfit

Name	Source	Category	Asymmetry	Border	Color	TDS
012383HB.jpeg	DermQuest	Benign Keratosis	2	5	3.0	4.60
012824HB.JPG	DermQuest	Benign Keratosis	0	2	3.0	1.70
019462HB.JPG	DermQuest	Malignant Melanoma	2	4	3.0	4.50
019475HB.JPG	DermQuest	Malignant Melanoma	1	2	3.0	3.00
019523HB.JPG	DermQuest	Malignant Melanoma	2	2	6.0	5.80
019563HB.JPG	DermQuest	Malignant Melanoma	2	1	4.0	4.70
019578HB.JPG	DermQuest	Malignant Melanoma	2	2	2.0	3.80
019681HB.JPG	DermQuest	Malignant Melanoma	1	4	3.0	3.20
019682HB.JPG	DermQuest	Malignant Melanoma	1	5	3.0	3.30
019725HB.JPG	DermQuest	Malignant Melanoma	0	2	4.0	2.20
019736HB.JPG	DermQuest	Malignant Melanoma	1	1	3.0	2.90
020192HB.JPG	DermQuest	Malignant Melanoma	1	5	4.0	3.80
020250HB.JPG	DermQuest	Malignant Melanoma	2	3	3.0	4.40
020266HB.JPG	DermQuest	Malignant Melanoma	1	0	3.0	2.80
020268HB.JPG	DermQuest	Malignant Melanoma	1	1	4.0	3.40
020302HB.JPG	DermQuest	Malignant Melanoma	2	8	4.0	5.40
020319HB.JPG	DermQuest	Malignant Melanoma	1	2	5.0	4.00
044936HB.JPG	DermQuest	Malignant Melanoma	2	3	4.0	4.90
005613HB.jpeg	DermQuest	Melanocytic Nevus	0	3	3.0	1.80
005728HB.jpeg	DermQuest	Melanocytic Nevus	2	3	4.0	4.90
005859HB.jpeg	DermQuest	Melanocytic Nevus	0	0	3.0	1.50
005900HB.jpeg	DermQuest	Melanocytic Nevus	0	3	5.0	2.80
006027HB.jpeg	DermQuest	Melanocytic Nevus	2	8	3.0	4.90
006074HB.jpeg	DermQuest	Melanocytic Nevus	1	6	3.0	3.40
006095HB.jpeg	DermQuest	Melanocytic Nevus	0	1	4.0	2.10
010769HB.jpeg	DermQuest	Melanocytic Nevus	2	6	3.0	4.70
010992VB.jpeg	DermQuest	Melanocytic Nevus	0	0	3.0	1.50

011031HB.jpeg	DermQuest	Melanocytic Nevus	0	0	4.0	2.00
011040VB.jpeg	DermQuest	Melanocytic Nevus	0	0	3.0	1.50
011201HB.jpeg	DermQuest	Melanocytic Nevus	1	0	3.0	2.80
011548HB.jpeg	DermQuest	Melanocytic Nevus	2	1	3.0	4.20
011697HB.jpeg	DermQuest	Melanocytic Nevus	1	4	4.0	3.70
011697HB.JPG	DermQuest	Melanocytic Nevus	1	4	4.0	3.70
011936HB.jpeg	DermQuest	Melanocytic Nevus	1	6	3.0	3.40
011936HB.JPG	DermQuest	Melanocytic Nevus	1	6	3.0	3.40
021733HB.jpeg	DermQuest	Melanocytic Nevus	1	3	1.0	2.10
021835HB.jpeg	DermQuest	Melanocytic Nevus	2	2	4.0	4.80
021837HB.jpeg	DermQuest	Melanocytic Nevus	1	0	4.0	3.30
021848HB.jpeg	DermQuest	Melanocytic Nevus	2	7	3.0	4.80
021852HB.jpeg	DermQuest	Melanocytic Nevus	0	0	3.0	1.50
021869HB.jpeg	DermQuest	Melanocytic Nevus	1	0	4.0	3.30
021878HB.jpeg	DermQuest	Melanocytic Nevus	2	1	3.0	4.20
021902HB.jpeg	DermQuest	Melanocytic Nevus	0	4	3.0	1.90
045720HB.jpeg	DermQuest	Melanocytic Nevus	0	1	3.0	1.60
046342HB.jpeg	DermQuest	Melanocytic Nevus	2	1	3.0	4.20
046343HB.JPG	DermQuest	Melanocytic Nevus	1	6	4.0	3.90

Table 9.2: Results of TDS calculation for DermQuest

Name	Source	Category	Asymmetry	Border	Color	TDS
IMD015.bmp	PH2Dataset	Melanocytic Nevus	1	5	2.0	2.80
IMD016.bmp	PH2Dataset	Melanocytic Nevus	1	2	2.0	2.50
IMD018.bmp	PH2Dataset	Melanocytic Nevus	1	1	2.0	2.40
IMD020.bmp	PH2Dataset	Melanocytic Nevus	0	1	2.0	1.10
IMD027.bmp	PH2Dataset	Melanocytic Nevus	2	7	2.0	4.30
IMD038.bmp	PH2Dataset	Melanocytic Nevus	1	2	3.0	3.00
IMD039.bmp	PH2Dataset	Melanocytic Nevus	1	3	2.0	2.60
IMD043.bmp	PH2Dataset	Melanocytic Nevus	2	7	3.0	4.80
IMD045.bmp	PH2Dataset	Melanocytic Nevus	1	1	2.0	2.40
IMD050.bmp	PH2Dataset	Melanocytic Nevus	1	6	2.0	2.90
IMD078.bmp	PH2Dataset	Melanocytic Nevus	2	8	3.0	4.90
IMD103.bmp	PH2Dataset	Melanocytic Nevus	1	5	2.0	2.80
IMD105.bmp	PH2Dataset	Melanocytic Nevus	1	4	2.0	2.70
IMD107.bmp	PH2Dataset	Melanocytic Nevus	2	3	2.0	3.90
IMD132.bmp	PH2Dataset	Melanocytic Nevus	2	5	3.0	4.60
IMD133.bmp	PH2Dataset	Melanocytic Nevus	1	2	2.0	2.50
IMD134.bmp	PH2Dataset	Melanocytic Nevus	2	4	2.0	4.00
IMD137.bmp	PH2Dataset	Melanocytic Nevus	0	5	2.0	1.50
IMD139.bmp	PH2Dataset	Melanocytic Nevus	1	3	2.0	2.60
IMD140.bmp	PH2Dataset	Melanocytic Nevus	1	5	3.0	3.30
IMD142.bmp	PH2Dataset	Melanocytic Nevus	1	5	3.0	3.30
IMD143.bmp	PH2Dataset	Melanocytic Nevus	1	2	3.0	3.00

IMD144.bmp	PH2Dataset	Melanocytic Nevus	1	2	3.0	3.00
IMD156.bmp	PH2Dataset	Melanocytic Nevus	1	3	2.0	2.60
IMD171.bmp	PH2Dataset	Melanocytic Nevus	1	4	2.0	2.70
IMD173.bmp	PH2Dataset	Melanocytic Nevus	0	7	2.0	1.70
IMD204.bmp	PH2Dataset	Melanocytic Nevus	0	1	2.0	1.10
IMD243.bmp	PH2Dataset	Melanocytic Nevus	1	8	3.0	3.60
IMD256.bmp	PH2Dataset	Melanocytic Nevus	2	7	2.0	4.30
IMD280.bmp	PH2Dataset	Melanocytic Nevus	1	2	2.0	2.50
IMD328.bmp	PH2Dataset	Melanocytic Nevus	2	1	3.0	4.20
IMD331.bmp	PH2Dataset	Melanocytic Nevus	1	5	3.0	3.30
IMD356.bmp	PH2Dataset	Melanocytic Nevus	2	2	3.0	4.30
IMD360.bmp	PH2Dataset	Melanocytic Nevus	1	1	3.0	2.90
IMD369.bmp	PH2Dataset	Melanocytic Nevus	2	5	3.0	4.60
IMD380.bmp	PH2Dataset	Melanocytic Nevus	1	3	2.0	2.60
IMD382.bmp	PH2Dataset	Melanocytic Nevus	2	4	4.0	5.00
IMD383.bmp	PH2Dataset	Melanocytic Nevus	0	3	2.0	1.30
IMD384.bmp	PH2Dataset	Melanocytic Nevus	2	5	2.0	4.10
IMD392.bmp	PH2Dataset	Melanocytic Nevus	1	6	2.0	2.90
IMD430.bmp	PH2Dataset	Melanocytic Nevus	1	2	3.0	3.00
IMD433.bmp	PH2Dataset	Melanocytic Nevus	2	4	3.0	4.50

Table 9.3: Results of TDS calculation for PH2Dataset