university of groningen

faculty of mathematics and natural sciences

COMPARISON OF FEATURES USED IN AUTOMATIC SKIN LESION CLASSIFICATION

SANDER FERINGA

Supervisors:
DR. ALEXANDRU C. TELEA
DR. MICHAEL H.F. WILKINSON
Advisor:
PAULO E. RAUBER

Master's Thesis Computational Science
University of Groningen

June 2015

Dedicated to my mother Coby Mulder,
who has recovered successfully from a melanoma in 1997.

ABSTRACT

Malignant skin lesions are an ever more common health problem in modern society. Certain types will even result in almost certain death when left untreated. The medical science community is therefore searching for better methods for diagnosing these lesions. Computing science can help doctors with this classification problem.

This work attempts to reveal how we can empower the designer of skin classification tools to effectively and efficiently explore the design space of skin lesion classification algorithms such as k Nearest Neighbours (kNN) and Support Vector Machines (SVM), by focusing on classifying birthmarks and melanoma.

After images have been segmented into healthy skin and skin lesion sections, a substantial group of descriptors are extracted from every segmented image. These include among others: common colour based features, statistical moments, Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), border features and co-occurrence matrix based descriptors. Feature vectors can be explored using the application *Featured*. It incorporates dimensionality reduction methods to generate 2D plots of the feature space. With the help of these plots we can explore the design space of descriptors and determine the influence of specific features, which in turn help us select high quality descriptors subsets for use in classifiers.

The highest classification accuracy score we achieved with our automatic classification system is 0.822, which is comparable to accuracy results attained by dermatologists. There are however still many aspects that influence the results negatively and therefore prevent the use of automatic classification systems in active medical service as an assistance tool.

ACKNOWLEDGMENTS

First I want to thank my supervisors Dr. Alexandru C. Telea and Dr. Michael H.F. Wilkinson for their advice and support. I also want to thank Paulo E. Rauber for his expansive knowledge on relevant subjects and support on some of the fine details. Regarding typography and layout, many thanks go to Elsbeth Bunschoten, Lucas Schucht, Laura Baakman, Anke Reinschlüssel and Jonathan Anketell for their help.

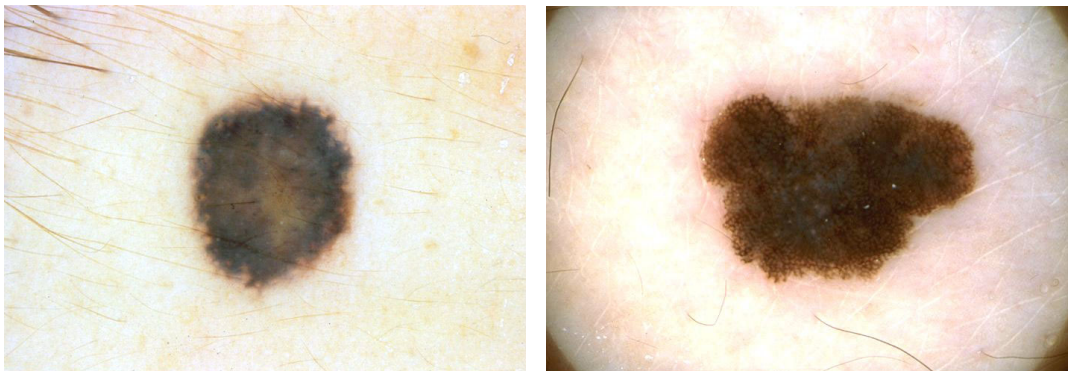
CONTENTS

1

INTRODUCTION

In our modern age of great scientific progress, the assistance of computer based systems has greatly advanced the capabilities of the medical world. However, the identification of malignant skin lesions has not been sufficiently advanced by this progress.

A dangerous class of pigmented skin lesions are the malignant melanoma: small lesions on the skin that are hard to distinguish from normal birth marks and are actually occurrences of malignant skin cancer. If such lesions are not identified in time and removed correctly, the patient's survival chance is slim. Figure 1 shows how difficult it can be to distinguish pigmented skin lesions. Since the number of new melanoma cases has increased significantly compared to other types of cancer [1], it is more important than ever that malignant skin lesions are detected in time.



(a) **(b)**

Figure 1: Examples of pigmented skin lesions. Figure (a) shows a benign blue naevus (a type of birthmark), figure (b) shows a malignant melanoma. At first glance it seems almost impossible to tell them apart.

Correctly classifying skin lesions seems to be an impossible task for the untrained eye. And although it is not an insurmountable exercise for medical specialists, it is still an effort met with difficulty. Certain types of lesions have characteristics that are easy to differentiate. However, several lesion types such as melanoma are hard to distinguish from other, sometimes harmless, types. It is often neither colour nor shape that can be seen as a unique feature for melanoma. The presence of specific details within a lesion cannot always result in correct classification of diseases. Multiple clinically and dermatoscopy based methods exist for the diagnosis of melanoma [2]. These include among others the *ABCD* and *ABCDE* rule-set, pat-

tern analysis and the *7 point system*. For example, based on the *ABCDE* rule-set a physician checks for the following characteristics [3]:

- **A**symmetry;
- **B**order irregularity or bleeding;
- **C**olour variation;
- **D**iameter > 6 mm;
- **E**levation.

Since medical specialists are just human, and humans make mistakes, it is likely that at some point errors are made in diagnoses. Cheng et al. state that: "In a recent study, general practitioners had a sensitivity and specificity for detection of melanoma of 62% and 63%, while dermatologists had a corresponding sensitivity and specificity of 80% and 60%." [4]. Therefore dermatologists are either as good or performing only slightly better as general practitioners and they are still substantially distanced from attaining near perfect classification accuracies. Furthermore, different medical doctors can give, hopefully only slightly, different diagnoses.

Several techniques and tools [5] can help physicians with the classification process. Most of these are however too unwieldy, too specific or too expensive for general practitioners to use. Their usage therefore remains limited to dermatologists. To help physicians either with the classification of skin lesions or to give patients a second opinion, big advancements made in computing science can help to create an automatic classification system for this purpose.

1.1 AUTOMATIC CLASSIFICATION

A substantial amount of research has already been done by other scientists to create such a system and almost all follow the same procedure. First the lesions are identified in digital images and separated from the rest of the image by a process known as segmentation. If needed the segmentations are pre-processed. This is followed with a feature extraction phase in which data with descriptive characteristics of the skin lesion morphology in the form of features (also known as descriptors) are extracted from the segmentation.

The feature data can then be used to determine the relevance of each individual feature. Features with low relevance can be removed to reduce the computational load in the following classification phase, which tries to classify the lesion based on the features. The data corresponding to new images is often compared with data gained from images of lesions that already have been classified. This procedure can be made into a general pipeline, such as the one presented in figure 2.

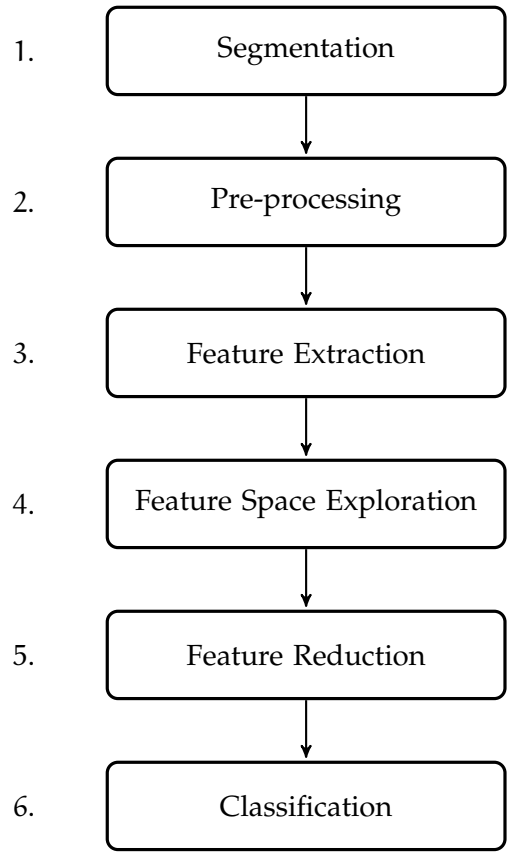


Figure 2: Pipeline of the automatic classification system stages.

1.2 DESIGNING HIGH QUALITY CLASSIFIERS

The creation of an automatic classifier is clearly not a simple matter. Each of the steps of the above specified general classification problem has numerous dependencies, factors and parameters that influence its outcome. This contributes to the many degrees of freedom available for designing a classification system with a high classification accuracy performance.

There are multiple methods to make segmentations, but which one is the best? What actually makes a good segmentation? There are many papers using only a smaller selection of descriptors, but why specifically those? Often there is no explanation given.

And considering classifiers: there are many methods available, but is there a classifier that always achieves the highest scores for the type of classification problem we are facing?

Given that there are so many degrees of freedom, designing a high quality classifier for a given set of images that capture specific skin morphology details is clearly not an easy task. The design space itself is high-dimensional and the designer

cannot easily explore it. Moreover, even if the designer had the time to exhaustively search this entire space, how would he or she be able to measure that a given parameter configuration is either better or worse than a different one? Therefore one of the major problems of designing a good classifier is providing good tools to the algorithm designer for exploring the design space.

1.3 RESEARCH QUESTION

From all these questions and unknowns, we can now formulate our research question:

> How can we empower the designer of skin lesion classification tools to effectively and efficiently explore the design space of skin classification algorithms in order to design better such tools?

Refining this query, we find two sub-questions:

1. Which are efficient and effective algorithmic building blocks for designing a skin classification pipeline that is effective in separating healthy skin from malignant skin tumours?

2. Which are efficient and effective tools for assembling the above pipelines from their components?

To answer these questions we go through the following chapters. In chapter 2, we discuss related work done by others on this field. Since the comparison of features is the most important goal of this thesis we give an extensive overview of the descriptors in the chapter 3, together with information on the tool used to generate the descriptors. Chapter 4 follows with information on classification methods and the corresponding tool used in this work. Chapters 3 and 4 answer our first research sub-question. The research itself is explained in chapter 5, including details of the datasets used and information on the exploration tool *Featured* made by Paulo Rauber [6]. This chapter also answers the second sub-question. All the results of the research are shown in chapter 6, followed by the discussion, future work and conclusion sections in respectively chapters 7, 8 and 9.

2

BACKGROUND INFORMATION

In this section we will discuss the work done by other researchers on related subjects. We will start with the medical background of melanoma, followed by a section on methods of image segmentation. The features used in this project are explained in detail in chapter 3, therefore we will only give a short overview on features in this chapter, together with a list of features that have not been included. After a general section on classification we will conclude with a description of several comparative works by others done before, including a look at the results they achieved.

2.1 MEDICAL BACKGROUND

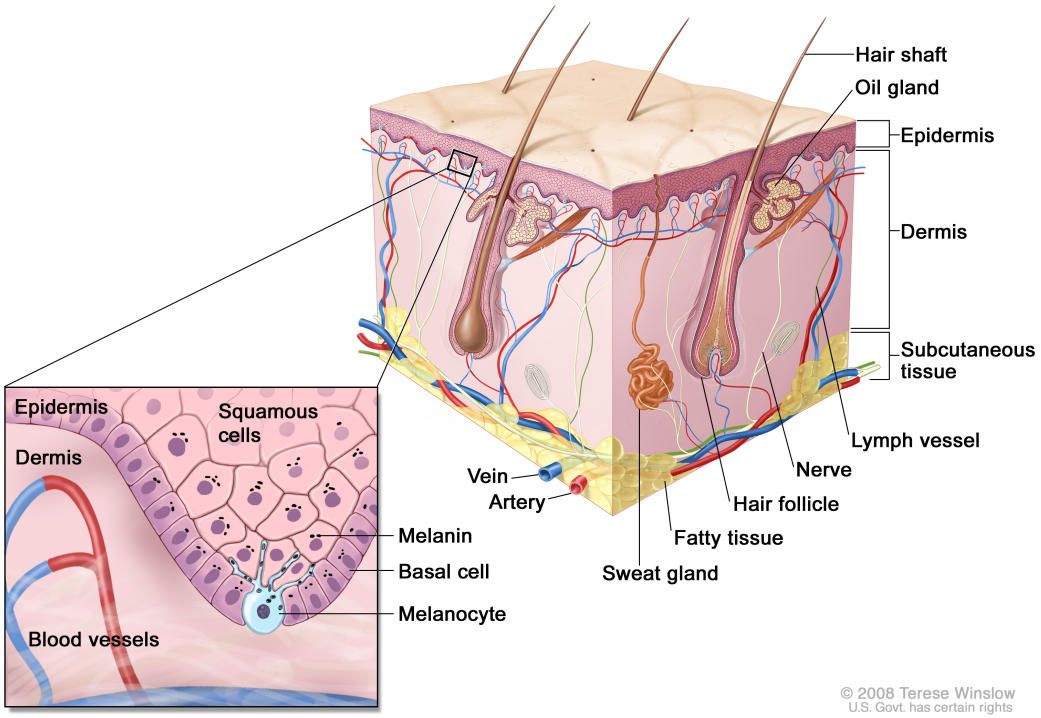


Figure 3: Anatomy of the human skin. The epidermis and dermis are shown, as well as the position of the melanocytes. Illustration "Melanoma Anatomy": For the National Cancer Institute © 2008 Terese Winslow, U.S. Govt. has certain rights

As explained by Korotkov [2], the human skin is a large and complex organ with two main layers: the epidermis and the dermis. In figure 3 we can see that the epidermis lays on top of the dermis. Keratinocytes are cells that, after their creation at the bottom basal layer of the epidermis, travel over the course of about 30 days to the horny layer at the top. Contained within the keratinocytes are packages of melanin pigment. The melanin pigment packages are made by the melanocytes, dendritic cells placed in the basal layer of the epidermis. The amount of pigment present in the skin cells controls the darkness of the human skin.

Like every other organ in the human body, cancer can also develop in skin cells. When non-pigmented cells develop skin cancer, they develop cancer types such as basal cell carcinoma and squamous cell carcinoma [7, 8]. Melanomas are created when cancer occurs in melanocytes.

Even though the number of melanocytes in the skin is substantially less compared to other cell types, melanoma cause 75% of all skin cancer deaths [2, 8]. In its final stages, melanomas are incurable and will almost certainly result in the death of the patient. In all stages before this phase, curing a melanoma is relatively easy: a large enough incision is often sufficient to remove the skin cancer completely.

Since melanomas are responsible for such a large portion of skin cancer deaths, many researchers have already invested in finding means to automatically classify melanoma. Automatic classification is made more difficult by naevi, a class of benign skin marks that are commonly known as birthmarks, since they share a strong resemblance to melanomas. 50% of all melanomas even grow out of pre-existing naevi [3]. Some people have a certain genetic predisposition that increases their changes for growing melanoma.

2.2 SOCIAL AND TECHNICAL BACKGROUND

The previous section has shown the importance of in-time and accurate detection of melanomas. Recognition of melanomas should be done when they are still in their early stages, either by medical experts or by patients spotting abnormal looking pigmented skin lesions and seeking professional help.

Manousaki et al. in [9] have stated that experienced dermatologists only have a classification accuracy of 64% to 80% using clinical diagnostic criteria. Even though these numbers are not low, the scores are still not high enough. Too many melanomas remain unnoticed or are noticed too late.

Although we do not believe that detecting melanomas should be done fully automatically and without the input of physicians, having a system that enables the general population to check their pigmented skin lesions to see if there is a risk of malignancy could be a possibility in the future. The assessment of the automatic

system can be a good reason for them to then visit a dermatologist for an actual test and classification.

Some extra technical details have to be explained before we continue with segmentation systems. We have used the term 'classification accuracy' or 'classification performance' before, but what accuracy or performance are we talking about? With accuracy or performance we mean the following: the number of correct classification predictions made, divided by the total number of classification predictions made in the form of a score in the range $[0, 1]$, or as a percentage.

There are several other metrics that can be used to express the quality of a classification system, such as precision, recall and the F-score. Since accuracy is well known and often used in literature, we will use it for our classification performance scores.

There are two types of images used in skin lesion classification systems. Besides the images taken by ordinary consumer or professional digital cameras, there are also images taken with specialized dermatoscopy devices. Modern dermatoscopes use polarised light to cancel out skin surface reflections and bring up certain details in the image that would otherwise remain unnoticed. Older dermatoscopes used non-polarised light and a special fluid applied on the skin's surface. These details coincide with the features included in the *ABCDE* rules list, shown in chapter 1.

2.3 SEGMENTATION SYSTEMS

Before classification can start, the skin lesion section has to be observed in the image and partitioned off (step 1 in figure 2). This is done to exclude details that are irrelevant for the actual lesion data we want to classify, since there is no guarantee that the features we use in the next step will not be negatively biased by such details. These details include surrounding healthy skin and hairs. Hairs that are on top of the lesion can be removed with a number of digital hair removal tools [10].

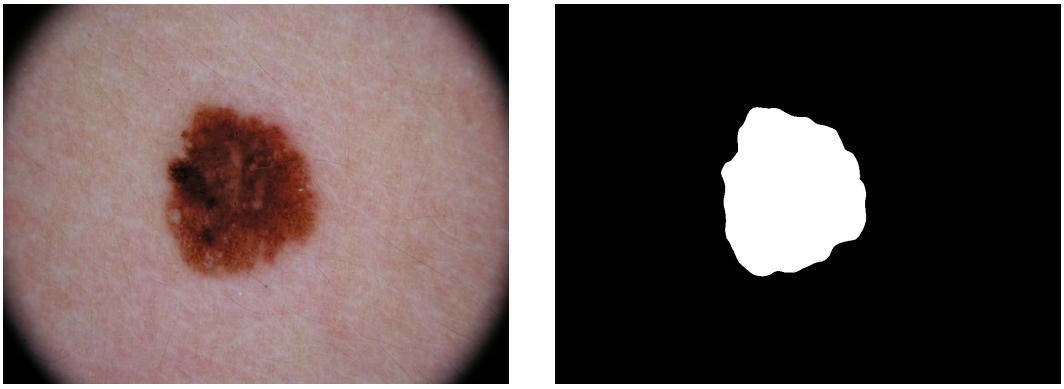
Automatic recognition of lesions can be a difficult task, since the difference in contrast between the lesion and healthy skin can be low. The border between the two parts might be fuzzy or complex shaped and lesions often consist of several types of textures.

Segmentation systems have been either proposed or implemented by others. These include works by Parolin et al. [11], Christensen et al., [12], Celebi et al., [13] and Korotkov et al. [2]. Throughout the years different methods have been tried and tested to accomplish automatic segmentation. One can segment using thresholding, level sets, morphological filters [12], normalised cuts, snakes [11], mean shift, skeletons or with dividing images up in patches and then classifying the patches using k Nearest Neighbours (kNN), Learning Vector Quantisation (LVQ) [14, 15]

or Support Vector Machines (SVM) [16]. Another method uses the Imaging Foresting Transform (IFT), also know as the superpixels method [17]. An overview of several of these segmentation methods is given by Telea in [18].

This project is going to use two of these methods. One is a method based on snakes called Gradient Vector Flow (GVF), as used by Parolin, Herzer and Jung in [11]. An optimised version of this method implemented on Graphics Processing Units (GPUs) in *NVidia's* Compute Unified Device Architecture (CUDA) platform exists and was implemented by Jans and Kiers in [19]. The other method is the superpixel method using the IFT by Rauber et al. in [17].

We will now give a short summary on their workings. For a detailed explanation of the GVF and superpixel techniques we have to refer to their original papers. An example image and mask can been seen in figures 4a and 4b.



(a) **(b)**

Figure 4: (a) Example image of an unclassified pigmented skin lesion, together with (b) its mask, which in this case has been created with Parolin's snakes based segmentation system [11].

2.3.1 *Gradient Vector Flow*

GVF is a variant of Active Contour Model (ACM), also known as snakes. ACM was introduced by Kass et al. in [20] and used in a medical setting before in [21] and [22]. ACM works by letting a curve 'shrink' according to a specific functional energy from its starting position until it has formed according to a shape present in a grey-scale version of the image. This is known as 'converging the snake'. The functional energy consists of internal and external energies. The internal energy represents the curve itself, the second part controls the external force, which consider the image data, guiding the curve to the boundary of the shape.

Xu et al. in [23] first proposed GVF and it solves a problem that the standard snakes implementation often had: poor convergence performance for concave boundaries

when the snake is initialised far from the minimum convergence state. This is done by exchanging the normal external force part of the energy function by a system based on a GVF field. Xu explains that GVFs:

> "... are dense vector fields derived from images by minimizing an energy functional in a variational framework. The minimization is achieved by solving a pair of decoupled linear partial differential equations which diffuses the gradient vectors of a gray-level or binary edge map computed from the image." [23]

2.3.2 *Superpixels*

The superpixel, or IFT method as explained by Rauber et al. in [17], is based on a totally different technique. Since the explanation of the method in the original paper is so concise and thorough, we will quote it:

> "Firstly, the input image is oversegmented. Seed pixels defined by the user associate a label to some of these oversegmented regions (superpixels). A superpixel graph is created to represent the oversegmentation: each superpixel corresponds to a node and arcs connect superpixels that are adjacent in the input image. An image foresting transform is then applied to associate a label to each superpixel, exploring the connection strength between superpixels and seeds." [17]

The last step can be seen as a competition to label every pixel in the image. When the user has set the seed points (markers) and applied the algorithm, performance is near real time on modern computers when working on images of moderate resolution.

2.3.3 *Comparing Gradient Vector Flow and Superpixel segmentation methods*

Also note that both the GVF and the superpixel method depend on non-automatic initialisation. Fully automatic segmentation techniques exist (such as the thresholding or the morphological filtering methods), but their segmentation accuracy is substantially lower according to their corresponding papers.

We have decided to use these two segmentation methods since they generally result in higher quality segmentations, are more robust and perform faster than any of the other methods. If we compare GVF with superpixel, we find that GVF produces slightly smoother contours and is less sensitive to noise such as hairs, while the superpixel method produces more accurate or detailed segmentations, which could have a positive influence on the features that need to be obtained in the next step.

2.4 FEATURE EXTRACTION

Using the segmented lesion as a base, we now need a system that can determine the skin lesion's type. Since it is hard for a computer to objectively 'look' at the lesion like a doctor would (if this would be possible we would have solved a difficult and persistent Artificial Intelligence (AI) problem), we need a system based on comparison. We can do this by extracting a number of identifying traits in the form of features (step 3 in figure 2). Features are also known as descriptors and they describe the the measurable aspects of an segmented image, which supports the automated quantified analyses of said segmented image.

Detailed overviews of features are given in [2, 13]. These features are divided into several main categories: colour, colour spaces, texture, boundary and an array of other descriptor types. Several are explained in detail in [11, 24, 25] among others.

Even though Maglogiannis et al. in [25] use the *ABCD* rule as a basis for their feature categories, descriptors used for an automatic system can generally not be mapped one-on-one to procedures used by humans. For example: the size of a lesion is often not a usable descriptor for an automatic classifier, since in many cases there is no scale or position information included with the digital image.

Maglogiannis [25] also talks about the use of dermal features such as skin elasticity, skin impedance, epidermis volume and epidermis thickness. These features are the image's metadata: data that cannot be extracted from an image of the lesion, but can only be gained from other known facts or measurements. Since most datasets do not contain such metadata, it is not possible for us to use these types of features.

Selecting appropriate descriptors for a classifier can be a difficult task. It is therefore strange that a substantial number comparable publications do not give reasons for selecting specific features. Even papers that do give an overview of features (such as Korotkov et al. [2]) only show which features are available, without going into their advantages or drawbacks. Chapter 3 will give a detailed overview on all descriptors used in this work. Several features that did not make the cut are explored there too.

2.5 CLASSIFICATION

And now the final step in figure 2: classification, which aims to assign a class label to an observation or element of a test set so that it matches as good as possible the manually assigned class labels by a professional. This is done by inferring class labels from measurable quantities, which in our case are the features.

For every disease type a substantial number of example segmentations are needed. These should be classified by professionals, preferable by multiple specialists who

agree with each other. The example segmentations are combined into a training set for use in a classification algorithm.

There are different types of classification algorithms. These include kNN, LVQ, logistic regression, Artificial Neural Networks (ANNs), decision trees and SVMs. A comparison of several classifiers is given by Dreiseitl et al. [26]. Their main conclusion is that the decision tree classifier is not adequate for skin lesion classification. kNN performs well and logistic regression, ANN and SVM exceptionally well.

Parolin et al. [11] use a dimensionality reduction step followed by a Bayesian classifier. Korotkov and Garcia [2] give an overview of work done by others with different classifiers. Maglogiannis and Doukas [25] compare classifiers types on the same dataset and set of features.

These conclusions are in agreement with the ones made by Dreiseitl et al. Korotkov adds that supervised machine learning algorithms generally perform better than unsupervised methods. Kusumoputro and Ariyanto [24] only use a Principal Component Analysis (PCA) dimensionality reduction step (even though they already had a small set of features) followed by a neural network classifier.

Since there are many different classifier types and their working can be quite complex, we will give an overview of common used classifiers:

- **k Nearest Neighbours**: works by calculating Euclidian distances in multidimensional space between the feature vector of an unlabelled observation and the feature vectors of all observations in the training set. The test observation is classified by doing a majority vote among the the k elements from the training set that have the shortest distance (and are therefore nearest) to the test observation. For more details, see chapter 4.

- **Learning Vector Quantisation**: This classifier can be seen as a optimisation step of kNN. This is achieved by reducing the amount of distance measures that need to be calculated. LVQ is a classification method based on prototypes of the data and was introduced by Teuvo Kohonen [27].

  Prototypes need to be chosen in such a way that each prototype is a good representation for one of the classes. Using prototypes, only the distances between the object and the prototypes have to be calculated. In the training phase the prototypes move closer to the datapoint belonging to the same class. When it belongs to a different class it is moved further away. After this procedure, the prototypes should be at optimum positions to represent their class. After a certain number of epochs the situation is sufficiently stable and the training phase is complete. Distance measures other than Euclidian distance can often be used to great effect [28, 29, 30, 15].

- **Logistic regression**: a simple progression from a threshold classifier. For a two class problem, logistic regression attempts to soften the threshold clas-

sifier's hard nature. Russel and Norvig [31] explain that it achieves this by interchanging the simple threshold function for a continuous and differentiable function: the logistic function. This function is defined as:

$$\text{Logistic}(z) = \frac{1}{1 + e^{-z}}. \tag{1}$$

The resulting value in the range $[0, 1]$ from this function corresponds to the probability of a sample belonging to either class.

- **Artificial Neural Network**: statistical learning algorithms based on biological neural networks like the human brain. The first preliminary mathematical work on ANNs was done by Warren McCulloch and Walter Pitts [32]. The first practical version was developed by Donald Hebb in the form of Hebbian learning [33].

  ANNs (see figure 5) are made with a network of interconnected nodes where one or more nodes are defined as input nodes and one or more defined as output nodes. All the other nodes are seen as hidden nodes and are ordered in one or more layers. The nodes represent neurons in biological neural networks.

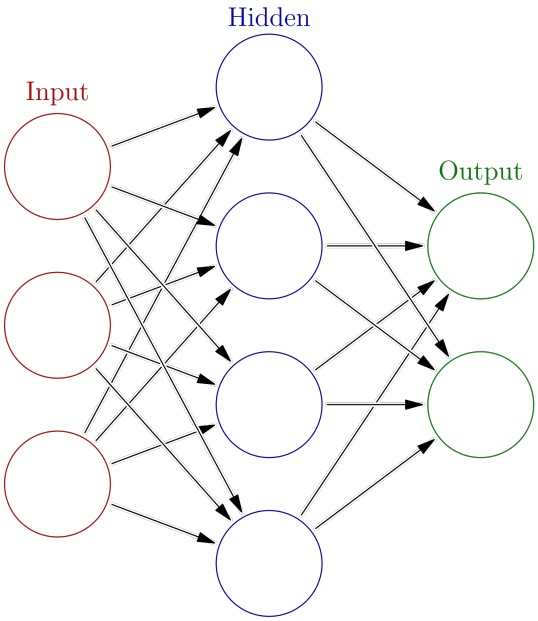


Figure 5: An ANN node diagram with three layers of nodes.

Quoting Russel and Norvig [31] on these nodes: "Roughly speaking, it 'fires' when a linear combination of its inputs exceeds some (hard or soft) threshold ...". A learning algorithm is used to train a set of adaptive weights spread out over the nodes. The adaptive weights correspond to connection strengths between neurons and can be activated during training and prediction.

ANNs have several advantages: since the network of nodes is inherently distributed in nature, it is easy to implement a distributed version. The network is also capable of approximating non-linear functions of their inputs. This is an advantage in many cases, since most real life situations behave non-linearly. ANNs also show graceful degradation when noisy input data is present. Even if several nodes are removed from the network, an ANN still functions for the greater part.

Suykens et al. state that ANNs can avoid the curse of dimensionality since: "... the approximation error becomes independent of the dimension of the input space." [34]. And they also state that ANNs: "... will be able to better handle larger dimensional input spaces than polynomial expansions, which is an interesting property towards many real-life problems where one has to model dependencies between several variables." [34].

According to Tu [35], ANNs also have several disadvantages. One of them being their 'black box' nature. Compared to other classifiers it is hard to determine which nodes and weights in the network have the highest contribution to a particular output. Therefore, a creator of an ANN cannot have a complete understanding of which parts of the network map to which section of the modelled relationship.

ANNs are also prone to overfitting, a problem that occurs when models are too complex and describe random error or noise instead of the underlying relationships. The training phase of the ANN can also take a substantial amount of time.

- **Bayesian classifier**: also known as an naive Bayes model is a classifier based on probabilities [11, 31]. Given two classes w_1 and w_2 representing naevus and melanoma respectively, the classifier will attempt to determine the posterior probability that a feature vector x belongs to a class w_i based on the Bayesian decision rule:

$$P(w_i|x) = \frac{p(x|w_i)P(w_i)}{p(x)}. \tag{2}$$

  Here $p(x|w_i)$ is the probability density function for w_i and $p(x)$ the probability density function for all x. Parolin et al. [11] and Duda et al. [36] use a cost function to calculate the cost that is attached to wrong selections. According to Parolin, the fact that the Bayesian classifier works with probabilities: "... allows a greater flexibility towards the results." [11].

  When a naive Bayes model is used as a classifier it has the disadvantage that it has strong feature independence assumptions. As in: it assumes that features are independent given the class label, and this is not always the case in classification problems like ours.

- **Decision trees**: a tree based structure where each branch of the decision tree represents a possible decision or occurrence. It consists of a sequences

of tests for each inner tree node, resulting in a decision which determines the direction within the tree that needs to be taken [31]. Each interior node corresponds to one of the input values. Edges from parent to children node are present for each of the possible values (ranges) of that input variable. Each leaf represents a value (range) of the target variable.

There are decision tree learners available that can generate a decision tree from a given problem. A disadvantages of this classifier is that learning an optimal decision tree is known to be computationally expensive, since it is a NP-complete problem.

- **Support Vector Machine**: a binary SVM is a type of linear separator that attempts to separate two classes of observations with a single straight line. There are however an infinite number of lines that will accomplish this task. The SVM tries to find the optimum line for this separation: the 'maximum-margin' line, or the line that is 'most in the middle'.

  In many datasets however, it is not possible to find a straight line to accomplish the separation. Instead of using a curved line to achieve separation, more advanced SVMs solve this problem by 'lifting' the features of the observations to a higher dimension using kernel functions. Therefore creating a hyperplane instead of a curved line, which can be solved by the default linear SVM. For more details, see chapter 4.

We will be using kNN and SVM classifiers in our research. kNN is easy to implement and often seen as the 'default' option. Since only testing with one type of classifier is insufficient, we are also using the SVM classifier in two variants: Linear and Radial Basis Function (RBF). kNN and SVM can be seen as opposite sides of the current field of classifiers. While one is simple, the other is complex. SVM is known to perform well in many classification situations and is therefore a good candidate for a complex classifier.

2.6 CLASSIFIER LIMITATIONS

Several of the previously summarised literature mention classification scores. Parolin et al. in [11] achieve an accuracy of 88.41% combined with a false negative score of 11.47%. Manousaki et al. [9] reach a sensitivity score of 60.9% and a specificity score of 95.4%, resulting in an accuracy score of 89.4%. In [25] Maglogiannis and Doukas give an overview of a range of classifiers as their results, ranging in accuracy score for the classification of naevi and melanoma from 95% to 100%. These last scores seem exceptionally high.

Kusumoputro and Ariyanto achieve a high accuracy score of almost 92%. However, they do this on a small dataset of only 63 images and on a small set of features, therefore the relevance of the results is somewhat questionable. Celebi et al. in

[13] come to a final score on a 564 image dataset of 92.34% specificity and 93.33% sensitivity.

Another example can be found in [37] by Elbaum et al. Here they achieve high sensitivity (95 - 100%) and moderately high specificity (68 - 85%) scores. However, there are some extra remarks that have to be made regarding their work: they use a special dataset that has 10 grey-level images for every skin lesion in the dataset, each taken with a different colour spectrum (including infra-red). Their dataset is highly asymmetrical: 63 melanomas compared to 183 melanocytic naevi. Finally it seems that they have had a big influence on the picture taking process when constructing their classification system. This has a major advantage on the classification outcome in a positive sense. However, it does give an indication of the results attainable by an automatic classification system in a highly controlled environment.

Overall we can conclude that there is substantial spread in classification results in existing literature. Which suggests that there are just too many existing parameters that influence the classification outcome. This also makes it impossible to do a definitive prediction on the outcome of our system. Parameters with major influence include: the quality of the dataset (in particular the number of images, the quality of the images, the consistency of the lighting conditions and the asymmetry of the dataset), the set of features used, the type of classifier used and the method of result representation (such as accuracy, sensitivity and specificity or f-score).

But when Cheng states that they can achieve an automatic classification: "... with a successful classification rate of 86% for detecting malignant melanoma. This is comparable with the clinical accuracy of dermatologists." [4] we have to conclude that it is possible to create an automatic classification system which achieves a classification accuracy consistent with the performance of dermatologists. An automatic classifier can therefore be a viable tool in the professional medical world. A section in chapter 7 is dedicated to comparing our accuracy scores with some of the results from the research mentioned above.

3

FEATURE GENERATION TOOL

There are multiple phases that a skin lesion image goes through before it is actually (hopefully correctly) classified. The first phase is the segmentation step as explained in the previous chapter, followed by the extraction of the descriptors themselves. The feature extraction we propose is implemented in a standalone command line tool called Feature Generation Tool (FGT), or `featuregen`. Its workings will be explained later in this chapter, after an extensive overview of all the features used by the tool, together with certain techniques needed by the descriptors. We end with a discussion section on the features and the tool itself.

3.1 SEGMENTATION MASKS

The segmentation phase (step 1 in figure 2) discussed in chapter 2 finished with binary masks for all images in the dataset. Healthy skin in these mask images is stored as black pixels with corresponding value 0. The segmented skin lesion area is stored as white pixels, which are values in the range $[1, 255]$. When the FGT creates binary mask images, it treats all values > 0 as white pixels and automatically changes them all to 255. This has the advantage that when a user wants to manually check mask images, image viewing tools show the lesion as white and not as an almost completely black colour, which would happen if the values stayed close to 0.

As we shall later explain, there are several descriptors that use the border of the skin lesion. This is a one pixel wide line of outermost skin lesion area pixels. However, datasets can contain images where the lesion border touches or intersects the image border. When an intersection with the image border occurs, the shape of the lesion border does not completely represent the shape of the actual lesion any more. Generation of boundary based descriptors in the tool can be changed by the user if necessary.

When the tool is used for other classification purposes in which there is no mask present or needed, a default mask is generated for every input image of corresponding size. Evidently, the border features are not computed in this case.

3.2 PRE-PROCESSING

Since the images in the datasets are taken with different types of cameras and under different lighting conditions we cannot guarantee image quality consistency. Images can be slightly blurred, or have low contrast. All these factors can have a negative effect on classification accuracy.

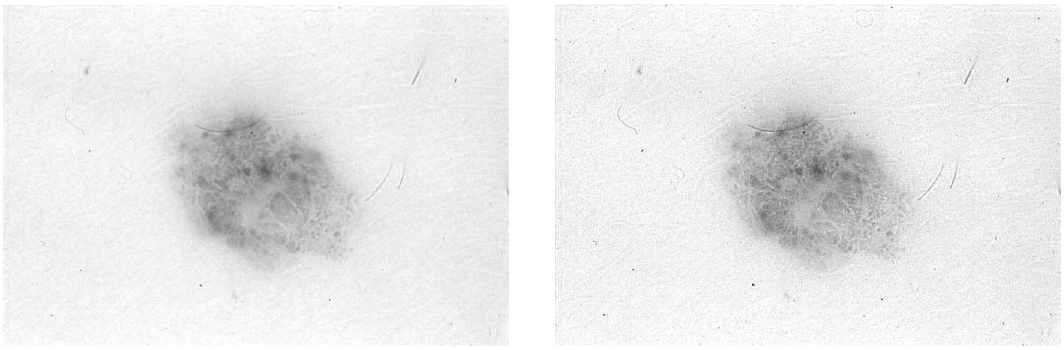
Some of the negative aspects of the input images can be counter-acted by pre-processing effects. These are applied on the original image before the features are extracted (step 2 in figure 2). Jung and Scharcanski's method [38] is an example for a pre-processing stage in a skin lesion classification system. It was also used by Parolin et al. [11]. To test the influence of these pre-processing steps, two filters are tested: unsharp mask and contrast stretching.

Unsharp mask is a sharpening technique often used by photographers to sharpen the often slightly blurred images that come out of standard compact and professional cameras. As explained in [39], a blurred version of the image is made using a Gaussian blur operation. This blurred version is subtracted from the original to form a mask image of edges. This mask is added to the original to form a sharpened version of the original image. The effect can be controlled by the type of Gaussian blur used and by a multiplier during the final addition process. An example of the application of the filter is shown in figure 6.

Contrast stretching is applied to low contrast images. For example: if a colour channel of an image has the range $[0, 255]$, but all the actual colour values of the image are in a smaller range, such as $[70, 130]$, we would like to increase the contrast of the image by using the full range $[0, 255]$. With contrast stretching, all the values between the measured minimum and maximum values of an image in a colour channel (here 70 and 130 respectively) are stretched out over the complete range of the channel. Contrast stretching can therefore be seen as a scaling normalisation process applied to the values of each channel of the image [39]. For an example application of the filter, see figure 7.

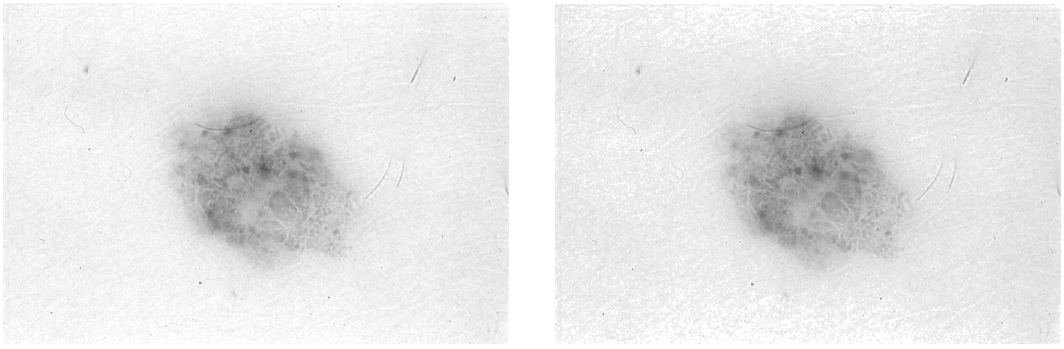
3.3 FEATURE EXTRACTION

We will now focus on the selection of features that are extracted for our implementation in step 3 of figure 2. Many types of features exist, spread over different classes. For this thesis we will focus on several features from colour channels using two colour spaces. Several features use a co-occurrence matrix of the Red, Green and Blue (RGB) colour space image. There are also features included that are based on the boundary shape of the skin lesion and a few texture-based descriptors. Since we obviously cannot test every existing feature in a single thesis, a selection was made



(a) **(b)**

Figure 6: Original input image (a) and output image (b) of an unsharp mask filter, applied to the red colour channel.



(a) **(b)**

Figure 7: Original input image (a) and output image (b) of a contrast stretching filter, applied to the red colour channel. The visual effect of applying this filter is quite limited, even though the values of most pixels will have changed.

based on usefulness of descriptors as shown in literature, ease of implementation and avoiding duplication of function.

A feature can be seen as a function from image space to some space R^n, where $n \geqslant 1$. For example: $n = 1$ when we calculate a mean for one colour channel, but $n = 16$ if we create a histogram with 16 bins (we will explain these features later in this chapter).

Using our batch processing tool we will calculate each feature for every image. The output of all descriptors on one image is combined into a single feature vector. Since we have a substantial number of features (of which some use histograms), the dimensionality of the feature vector rises into the hundreds. All the feature vectors are combined with image file names and attribute labels into one single text file so it can be used by other tools. The feature vector will be used by both the exploration tool *Featured* to help examine the usefulness of each feature and by the classification tool later on to classify the skin lesions.

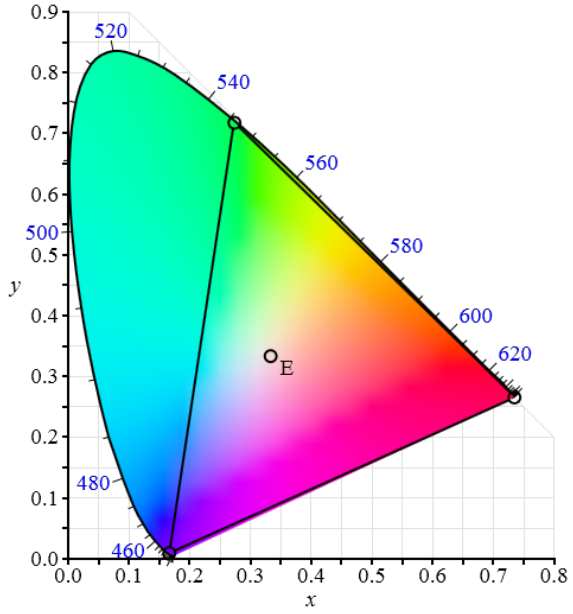


Figure 8: Comparison of the Lab horseshoe and the RGB triangle. Image by Wikipedia user BenRG, public domain license.

3.4 COLOUR SPACES

Colour images can be represented using different colour spaces. Each space has its own advantages and disadvantages. Both of the following spaces have been developed by the Commision Internationale de l'Eclairage (CIE). All pictures we use as input for our tool-chain are in the RGB format. This can actually be the *Adobe RGB* or the *sRGB* format, both are absolute colour spaces and implementations of the RGB colour model, but since they only differ in minute details we will not go into further detail here.

Besides RGB we will also look at the CIE 1976 colour space. Also known as *CIELAB*, or Lightness, *a* colour component and *b* colour component (Lab) colour space, this colour space contains all luminance information in one channel and all the colour information in two other channels and is therefore very useful for this application. A comparison of both colour spaces can be seen in figure 8. We use these two colour spaces because certain details or variations in image patterns may be better distinguishable in specific spaces.

3.4.1 *RGB*

The RGB colour space is used in many digital cameras and computer displays. Colours are produced by adding red, green and blue values in different proportions.

Luminance is included in each channel separately and not as a separate channel. Using 8 bit values for each primary colour in the range $[0, 255]$ there are a total of $256^3 = 16.7 \cdot 10^6$ possibilities. Using 8 bits per channel is the default option in contemporary digital photography, but not a requirement. Many professional cameras use 12, 14 or even 16 bit channels in their own Raw Image Format (RAW) file formats. However, since many systems, including monitors and TVs cannot handle anything other than 8 bit, the images are often reduced to 8 bit per channel images as a post-processing operation. This processing happens either internally in the camera or on personal computers in image editing tools.

The RGB colour space is a subset of all the colours humans can see. As can be seen in figure 8, the RGB triangle is smaller than the horseshoe shape of the full human colour spectrum. On the other hand, RGB is a logical system to measure colour, since human eyes work with cones that are sensitive for red, green and blue light. This does not mean that humans do necessarily perceive colours in such a way in their mental representation of the world.

3.4.2 *Lab*

The Lab colour space is derived from the non-linearly compressed *CIEXYZ* colour space coordinates and it comes in several versions. We use the *CIE*1976*Lab* colour space version. The colour space *CIE*1931*XYZ* defines all the colours within reach of human perception. The intention of Lab is to be a colour space which can be computed simply from the *XYZ* space, but at the same time also be more perceptually uniform than *XYZ*. Perceptual uniformity in this case means that a change in colour value should produce a change of roughly the same visual importance.

3.5 FEATURES USING COLOUR CHANNELS

The descriptors are applied within the masked area on each of the colour channels. Several features such as the statistical moments, homogeneity, correlation, contrast, uniformity and entropy were first calculated over the histogram of each colour channel. This resulted however in a large number of bins staying empty in the histogram. Therefore these features need to be calculated over all the data. Unless stated otherwise, all these features are calculated in both RGB and Lab colour space versions of the images.

3.5.1 *Histograms*

A histogram is a representation of the distribution of data into frequencies for a pre-defined number of discrete intervals called bins. It was introduced by Karl

Pearson in 1895 [40]. Bins are equally distributed over the original data's range. In a standard 8 bit grey value image with a range of values of 256, the number of bins for a histogram of this grey value image must lay between 1 and 256. If there would be only one bin, the frequency of this bin will account for the total number of pixels in the image. If there would be 256 bins, we would have a unique bin for every possible kind of value in the image. m_i corresponds to the count of a histogram bin and is defined as:

$$m_i = \{p \in D(\mathcal{I}) \mid I(p) \in r_i\}. \tag{3}$$

Here D denotes the domain of the image $\mathcal{I}$, p denotes the position of a pixel of $\mathcal{I}$, $I(p)$ denotes the value of a pixel of $\mathcal{I}$ and r_i denotes the interval corresponding to the bin i. The histogram M_i is then defined as:

$$M_i = \frac{|m_i|}{\sum_j^n |m_j|}, \tag{4}$$

where we take the sum over the n counts of histogram bins. To preserve all the data, a histogram of 256 bins per colour channel must be used for a 8 bit colour image. But, since the skin lesion images in general have colour values that fall within the often occurring specific sub-ranges, a lot of those 256 bins will be 0 in the histogram. This would result in a large feature vector where most features have no contribution in describing the image.

There is however another reason why we do not want to use 256 bins. If there is noise, or if the lighting is slightly different or even if the patient has a different natural skin colour, the colour values are already different and will therefore result in a different histogram. We are not going after exact pixel colour values, but actually looking at small ranges of colours is useful.

To achieve this, we reduce the number of bins to hold for instance 8 (so 32 bins) or 16 (so 1 bins) possible colour value types. We could reduce the number of bins to be substantially lower than 16, but this would degrade precision too much. The number of bins can be specified in the tool and will be used for all the histograms in the features calculated by application.

Since every bin of the histogram is a count of pixels in an image there is a correspondence between the histogram of an image and the size, or pixel resolution, of the image. To remove this correspondence, every histogram created is normalised. Features that do not use a histogram will be included without normalisation to enable the option of using different normalisation techniques within the classification tool. The normalisation for histograms will reduce the range to $[0, 1]$.

The histogram of a colour channel exists as a separate descriptor in our set of features. Histograms are created according to a specified numbers of bins of all the colour channels in the RGB and Lab colour spaces. Several other features in the following sections also make use of a histogram as their descriptor output.

3.5.2 *Mean and standard deviation*

Multiple features are derived from the four standardised moments used in statistics. The moments form part of a systematic approach to distinguish probability distributions. They are powerful in terms of their mathematical and computational simplicity. The 0^{th} moment is the total probability and its mean is the first moment. The standard deviation (in our case the population standard deviation) of a vector details how the elements in a vector is spread out. It is the square root of the second moment, also known as the variance. For every colour channel we calculate the mean μ and standard deviation σ of an image $\mathcal{I}$ for all the pixels intensities $I(p)$ in the region of interest R, where $R \subseteq \mathcal{I}$. μ and σ:

$$\mu = \frac{1}{|R|} \sum_{p \in R} (I(p), \tag{5}$$

$$\sigma = \sqrt{\frac{1}{|R|} \sum_{p \in R} (I(p) - \mu)^2}. \tag{6}$$

The usefulness of the mean descriptor can be quite limited, since it is influenced greatly by the overall image quality, colour reproduction, gamma and other image characteristics. If we for instance would take multiple images of the same skin lesion with different cameras and lighting conditions, it is likely that non will have the same output mean value. This problem can also arise with the standard deviation feature, although its effect is less since this feature also helps define contrast differences within the lesions surface. A high σ value will correspond to a high contrast detail within the lesion, and therefore be a good descriptor for lesion classification. The standard range of a mean value of a 8 bit image is $[0, 255]$. Which corresponds to a theoretical limit of half that range for the standard deviation: $[0, 127]$.

3.5.3 *Variance*

The second moment is the averaged squared difference of the mean as calculated for all the pixels intensities $I(p)$ of the pixels p in the region of interest R:

$$\sigma^2 = \frac{1}{|R|} \sum_{p \in R} (I(p) - \mu)^2. \tag{7}$$

The paper by Parolin, Herzer and Jung [11] states that malignant skin cancers are characterised by darker spots of tan, red, brown and black compared to benign skin cancers. They therefore will have a higher average colour variance in the separate RGB colour channels. Taking squared values instead of the plain values makes it easier to use in algebra and removes all the negative signs. This is helpful since

we are not interested in the direction of the differences. As stated earlier, variance σ^2 is the standard deviation σ squared. The variances range is from 0 to 127^2, or $[0, 16129]$.

3.5.4 *Skewness*

The third moment, known as the moment coefficient of skewness, defines the asymmetric nature of the distribution of the data and can be calculated for all the pixels intensities $I(p)$ of the pixels p in the region of interest R using:

$$Skew = \frac{\frac{1}{|R|}\sum_{p \in R}(I(p) - \mu)^3}{(\frac{1}{|R|}\sum_{p \in R}(I(p) - \mu)^2)^{\frac{3}{2}}}. \tag{8}$$

In an unimodal graph (a graph with one peak) of a distribution, skewness indicates whether the tail of the graph is longer on one side of the peak or on the other. If skewness is positive, the data is positively skewed or skewed right, meaning that the right tail of the distribution is longer than the left. If skewness is negative, the data is negatively skewed, as in: skewed to the left. In this case the left tail is longer than the right tail. If the skewness is 0, the data is perfectly symmetrical. This is true for both unimodal and multimodal distributions.

The practical range for skewness is hard to determine. It is a generally accepted aspect in statistics that there is a significant skewness if a skewness value lies outside the range $[-1, 1]$. However, from the equation we can deduce that the feature's range in theory is $[-\infty, \infty]$. Since we cannot use ranges with ∞'s for normalisation, we have to use a more practical range. We have empirically determined that a range of $[-32, 32]$ is an adequate assumption.

3.5.5 *Kurtosis*

The fourth moment, or the moment coefficient of kurtosis, defines the height and sharpness of the peak in the graph of the distribution. If kurtosis is high, the peak of the distribution graph has a high and sharp shape. A low kurtosis value indicates that the peak is low and less distinct, making it look similar to rolling hills. A standard normal distribution has a kurtosis value of 3, while a uniform (square block shaped) distribution has a kurtosis of 1.8. The range of kurtosis is in theory $[1, \infty]$, but in practical situations there is a lower threshold. We have empirically observed that our images converge to a maximum of about 16, but because of safety reasons we will use 255 as an upper limit. The kurtosis can be

calculated with equation for all the pixels intensities $I(p)$ of the pixels p in the region of interest R:

$$\text{Kurt} = \frac{\frac{1}{|R|}\sum_{p\in R}(I(p)-\mu)^4}{(\frac{1}{|R|}\sum_{p\in R}(I(p)-\mu)^2)^2}. \tag{9}$$

3.5.6 *Relative Chromaticity*

This feature is included in a paper by Parolin, et al. [11] and in a paper by Kusumoputro and Ariyanto [24]. With relative chromaticity as defined by:

$$RC_c = \frac{\mu^c_{RGB\in L}}{\sum_{c\in\{R,G,B\}}\mu^c_{RGB\in L}} - \frac{\nu^c_{RGB\in H}}{\sum_{c\in\{R,G,B\}}\nu^c_{RGB\in H}}, \tag{10}$$

an attempt is made to use the difference in overall colour values between a part of the image J outside the skin lesion H and the area inside the skin lesion L.

L is generated by making two dilated version of the original mask images. The first version constitutes the original mask image dilated with a structuring element of 31 x 31 pixels in size. The second version is made in a similar fashion, but dilation is performed twice. By subtracting the first version from the second version a band shaped mask of 15 pixels wide is created which flows as a ring around the original mask at a distance of 15 pixels. Because the band would normally flow outside the original image, *OpenCV*s dilation implementation automatically removes the parts outside the original image. It can also handle cases where the original mask area touches or intersects with the image border.

The average colour value outside the lesion ν is subtracted from the average colour value inside the lesion μ. This is done for every colour channel c in $\{R, G, B\}$ of RGB. It is also stated by both papers about this feature that it will "reduce the small variation of lighting, printing and digitisation" [11, 24]. Practical values of the feature are often between -0.20 and 0.20. We will use a range of $[-1, 1]$.

3.5.7 *Colour Variance*

Besides relative chromaticity, colour variance is another feature presented in both Parolin's [11] and Kusumoputro's [24] papers. It compares colour variances instead of average colour values, for every colour channel c in $\{R, G, B\}$ on the image J. The part of the image outside the skin lesion is seen as healthy skin H and the area inside the skin lesion is described as L. For H we use the same area as described

in the relative chromaticity feature above. According to Kusumoputro, the colour variance of malignant skin is normally higher then the colour variance of healthy skin.

To calculate the variances, the same average colour value outside the lesion ν and average colour value inside the lesion μ are used. $I(p)$ denotes the intensity at a pixel p. As defined by:

$$CV_c = \frac{1}{|L|}\sum_{p\in L}(I(p)^c - \mu^c_{RGB\in L})^2 - \frac{1}{|H|}\sum_{p\in H}(I(p)^c - \nu^c_{RGB\in H})^2, \tag{11}$$

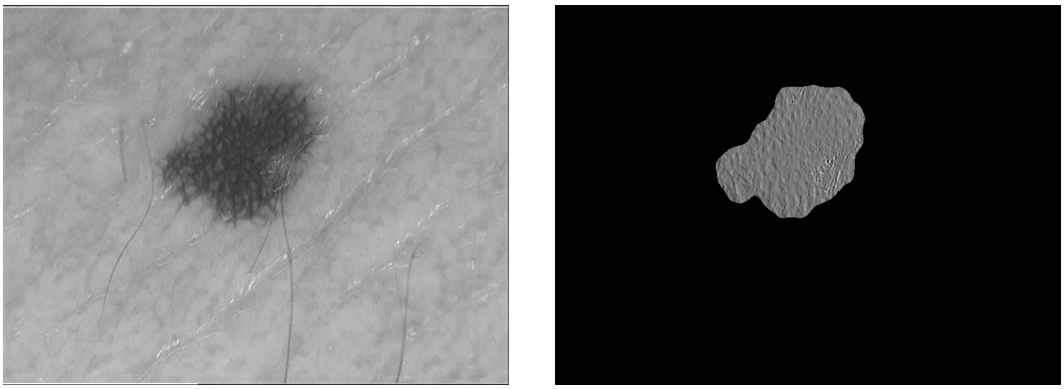
the variance of healthy skin is subtracted from the variance of malignant skin for every p of either L or H. Since the practical output values of this feature are very close to 0, we will again use a range of $[-1, 1]$.

3.5.8 *Local Binary Patterns*

The Local Binary Patterns (LBP) feature describes both colour and texture details and was first described by Ojala, Pietikäinen and Harwood in 1994 and later published in 1996 [41]. LBP is often used together with Histogram of Oriented Gradients (HOG) and SVM for recognising humans in images and videos. But it can also be a good descriptor on its own. The standard implementation of HOG follows the following steps:

1. The image is divided into cells (often 16 x 16 pixels for each cell).
2. Each pixel in a cell is compared with each of its 8 neighbours. These pixels are followed along a circle in a fixed direction, consistently clockwise or counter-clockwise.
3. If the centre pixels value is greater than the neighbours value, we store a 1. Otherwise, we will store a 0. These stored numbers are combined into a 8 digit binary number.
4. Compute the histogram over each cell of the frequency of each occurring "number" (the 8 digit binary number we generated in the previous step).
5. Optionally normalise the histogram.
6. Finally the histograms of all cells are concatenated. This results in the feature vector of the image.

For our purpose we want to use a variant of the standard LBP descriptor. Instead of using cells, we see the whole masked area of the skin lesion image as one cell. We do this since it is difficult to divide most of our masked areas into cells, and because we are not interested in differences between regions within the masked area. LBP



(a) **(b)**

Figure 9: Original blue colour channel inly image of a melanoma (a), and its counterpart (b) with the Sobel filter applied within the masked area.

with cells however would be a useful descriptor for a follow up study on the differences in regions within lesions and their influence on skin lesion classifiers. Another difference is that we do not keep the 8 digit binary number, but convert it to a regular number in the range $[0, 255]$ range. This way we can use existing implementations, such as the histogram, later on.

3.5.9 *Histogram of Sobel Edges*

This feature is both a RGB colour based descriptor and a texture descriptor, but it is not applied to Lab colour space. It describes edges within an image in such a way that they can inform us on lesions texture properties. The Sobel operator filter is only applied in RGB colour space and not in Lab colour space, since the two colour information channels of Lab do not hold details with high enough contrast. Contrast which the Sobel operator needs to find the edges. The L channel could be used, but it would lack the differentiating information given by the contrast details in RGBs three channels.

There is a visual difference in structures and textures between sick and healthy skin in skin lesion images, but there is also a difference in texture between different lesion types. Most edge detection algorithms generate an 1 bit image per colour channel as output, which for our purpose would result in too much information loss.

We use a special implementation of the Sobel edge detection from Trucco and Verri in [42] that results in three 8 bit grey value images (one for each colour channel). This implementation mimics the *Matlab* version as used by the author of this thesis [14, 15]. The final output of the feature is 3 histograms: one for each channel. As a bonus feature, we also add the means of G_x and G_y for every channel. The Sobel operator for an image $\mathcal{I}$, with a 2D convolution $\star$ and where G_x and G_y

are two images which at each point contain the horizontal and vertical derivative approximations:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \star \mathcal{I} \qquad \text{and} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \star \mathcal{I}.$$

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude image G, using:

$$G = \sqrt{G_x^2 + G_y^2}. \tag{12}$$

3.5.10 *RGB co-occurrence matrix based texture descriptors*

The following five features are from [39] and can be used as texture descriptors. They have their origin in a paper by Haralick [43] and they are all numerical features computed from the co-occurrence matrix of the skin lesion image area. The use of the co-occurrence matrix for these features comes from another paper by Haralick [44]. It is used to describe textures by comparing differences in pixel values in small local neighbourhoods by comparing its direct neighbours. A co-occurrence matrix C is defined over an M x N image $\mathcal{I}$, parameterised by an offset $(\Delta x, \Delta y)$ as seen in:

$$C_{\Delta x, \Delta y}(i, j) = \sum_{v=1}^{M} \sum_{w=1}^{N} \begin{cases} 1, & \text{if } \mathcal{I}(v, w) = i \text{ and } \mathcal{I}(v + \Delta x, w + \Delta y) = j, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The range of the colour channels determines the size of C: an 8 bit image has $2^8 = 256$ possible values per channel, making both the height and width K of the square co-occurrence matrix 256. It is common for Δx and Δy to compare the current pixel with one pixel to the right $(\Delta x + 1, \Delta y)$ and with one pixel up $(\Delta x, \Delta y + 1)$.

Comparing with one pixel down or one pixel to the left seems necessary at first, but since both the up and down directions give the same outcome, and the left and right directions give the same outcome, it is unnecessary.

This effect occurs because neighbouring pixels are opposites of each other in either the vertical or horizontal direction. Therefore the sum of all parts will still result in the same outcome for every metric that uses the co-occurrence matrix.

The lack of differentiating information in contrast details from the three Lab colour channels gives a reason for the co-occurrence based descriptor to be only applied to the RGB colour channels.

3.5.11 *Homogeneity*

According to Gonzales and Woods: "Homogeneity measures the spatial closeness of the distribution ..." [39] for a lesion image area. C denotes the square co-occurrence matrix of size K x K. p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C. $\mathrm{Hom}(C)$ is therefore defined as:

$$\mathrm{Hom}(C) = \sum_{i=1}^{K} \sum_{j=1}^{K} \frac{p_{ij}}{1 + |i - j|}. \tag{14}$$

A high homogeneity value Hom would mean that the matrix almost represents a diagonal matrix. A low homogeneity value would mean that the values differ substantially from a diagonal matrix. As in: no diagonal features would be present in the texture of the lesion. The range for homogeneity values is $[0, 1]$.

3.5.12 *Correlation*

The second co-occurrence matrix feature from [39] is correlation, defined as:

$$\mathrm{Corr}(C) = \sum_{i=1}^{K} \sum_{j=1}^{K} \frac{p_{ij}(i - cm_r)(j - cm_c)}{c\sigma_r c\sigma_c}. \tag{15}$$

It measures, for every pixel in the region of interest R, the correlation between a pixel and its neighbour. Its values have a range of $[-1, 1]$. Here -1 corresponds to a perfect negative correlation and 1 corresponds to a perfect positive correlation. cm_r, cm_c and $c\sigma_r$, $c\sigma_c$ denote the column mean and column standard deviations of the row and column sums of the current x_{ij} in the image:

$$cm_r = \sum_{i=1}^{K} i \sum_{j=1}^{K} p_{ij}, \qquad cm_c = \sum_{j=1}^{K} j \sum_{i=1}^{K} p_{ij}, \tag{16}$$

$$c\sigma_r = \sqrt{\sum_{i=1}^{K} (i - cm_r)^2 \sum_{j=1}^{K} p_{ij}}, \qquad c\sigma_r = \sqrt{\sum_{j=1}^{K} (j - cm_c)^2 \sum_{i=1}^{K} p_{ij}}. \tag{17}$$

The correlation Corr of the co-occurrence matrix C with size K x K is only defined when $c\sigma_r \neq 0$ and $c\sigma_c \neq 0$. p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C.

The more randomness is occurring in x, the closer the correlation is to 0. Since generally skin lesion images have fine details, small spread-out structures and noise, it is likely that there will be some randomness in the co-occurrence matrices and therefore correlation values close to 0.

3.5.13 *Contrast*

The third co-occurrence matrix feature from [39]. As seen in:

$$\mathtt{Contr}(C) = \sum_{i=1}^{K}\sum_{j=1}^{K} p_{ij}(i-j)^2, \tag{18}$$

contrast $\mathtt{Contr}$ measures the intensity contrast for each pixel related to its location in the image. C denotes the co-occurrence matrix with size K x K. p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C. The values of $\mathtt{Contr}$ range from 0 to $(K-1)^2$.

3.5.14 *Uniformity*

The uniformity $\mathtt{Unif}$, as defined by:

$$\mathtt{Unif}(C) = \sum_{i=1}^{K}\sum_{j=1}^{K} p_{ij}^2, \tag{19}$$

also called the energy or the 2^{nd} angular momentum, is another co-occurrence matrix co-occurrence matrixfeature from [39] with a range of $[0, 1]$. C denotes the co-occurrence matrix with size K x K. p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C. A value of 1 would indicate a completely uniform image.

3.5.15 *Entropy*

The last of the co-occurrence matrix features from [39] is entropy, or $\mathtt{Entr}$. It measures the randomness in the co-occurrence matrix C with size K x K. p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C. If al p_{ij}'s are 0, the entropy is 0. If all p_{ij}'s are at their maximum, the entropy will be its maximum value of $2\log_2 K$. $\mathtt{Entr}(C)$ is defined as:

$$\mathtt{Entr}(C) = -\sum_{i=1}^{K}\sum_{j=1}^{K} p_{ij}\log_2 p_{ij}. \tag{20}$$

3.5.16 *Absolute Value and Inverse Differences*

Materka and Strzelecki add two more features for the co-occurrence matrix C with size K x K in their review on texture analysis methods [45], they however do not give much information on these two features. The absolute value feature $Absval$ is

probably a close match to the contrast feature, as taking the square or the absolute of $i-j$ has the same effect. It is defined by:

$$Absval(C) = \sum_{i=1}^{K}\sum_{j=1}^{K} p_{ij}|i-j|, \tag{21}$$

where p_{ij} is the ij^{th} term of C, divided by the sum of the elements in C. Since the range of p_{ij} is $[0,1]$ and the range of $abs(255-0)$ is 255, the descriptors range is $[0,255]$

The range of the inverse difference feature equation is $[0,1]$ and it is defined as:

$$Invdiff(C) = \sum_{i=1}^{K}\sum_{j=1}^{K} \frac{p_{ij}}{1+(i-j)^2}. \tag{22}$$

3.6 FEATURES USING BOUNDARIES OR TEXTURES

The following features are not extracted purely from colour channels but use sources of different types. The corresponding mask of the image generated by the skin lesion segmentation stage is used to generate the boundary of a skin lesion image. Using the morphological dilation operator we generate a version of the mask that is 1 pixel larger in every direction. After subtracting the original mask we are left with a 1 pixel wide shape describing the boundary of the lesion.

HOG will always be executed. However, as explained before at the beginning of this chapter, only when we select the option to include boundary based descriptors the following features will be used: compactness, centroid difference, boundary moments, histogram of boundary to centroid distances and asymmetry index.

3.6.1 *Histogram of Oriented Gradients*

As first described by Navneet Dalad and Bill Triggs in [46], the descriptor HOG is often combined with a SVM to generate a system capable of detecting humans in images. The descriptor's name already implies its function: the creation of a histogram from the orientation of the gradients of the image area. To reduce computational load, we first convert the colour image to one grey value image. To get the gradient of each pixel, the same Sobel operator is applied. This is the same procedure as used in Sobel edge detection. However, instead of calculating the gradient magnitude, we now use the gradient orientation function Θ:

$$\Theta = arctan\frac{G_y}{G_x} * \frac{180}{\pi}, \tag{23}$$

applying it to every pixel of the image, resulting in an image containing the gradient Θ in degrees with a range of $[0,180]$ for each pixel. G_x and G_y denote the two

images which at each point contain the horizontal and vertical derivative approximations respectively.

Overall shapes in images with object detection are recognisable by local differences in gradients. Each images is therefore divided into cells which are usually 16 x 16 pixels each. The cells overlap each other by 50%. Since each cell only holds $16 * 16 = 256$ datapoints in total, we use a relatively low number of 9 bins to calculate a histogram for each cell. All histograms are concatenated into one large feature vector.

Because we are interested in overall lesion features and not in differences within the lesion, the HOG implementation is changed by referencing the whole masked lesion area as one cell. This results in one histogram in total as output instead of one per cell. The number of pixels is not much larger compared to individual cells. Therefore we have increased the number of bins to 16, down from the original gradient orientation range of 180 degrees. Since the number of pixels in each image is different, the values in the histogram are normalised to $[0, 1]$. This also applies to all other output histograms of descriptors.

3.6.2 *Compactness*

As used by Shen [47], this descriptor can be seen as a "... simple, dimensionless measure of shape compactness ..." [47] where the length of the boundary $|P|$ is squared and divided by the area of the region of interest (lesion) $|R|$ to attain the compactness $Comp$, or: the ratio between surface area and boundary length. Both $|P|$ and $|R|$ are in this case counted in pixels. The range for $Comp$ is substantially harder to determine. Since neither $|P|$ nor $|R|$ can be negative, $Comp$ can also never be negative. And by squaring the boundary length, $|P|$ and $|R|$ are within the same order of magnitude.

Skin lesions shapes come in different types: some are almost circular, some have a lot of creases, some consist of multiple smaller spots. It is safe to conclude that there will be no lesions with shapes that have such long boundary lengths compared to surface area that $Comp$ will be larger then 100. Therefore we conform us to a practical range of $[0, 100]$. The descriptor is mathematically determined by:

$$Comp = \frac{|P|^2}{|R|}. \tag{24}$$

3.6.3 *Centroid Difference*

The centroid difference is a simple feature for expressing the boundary shape of a lesion. There are two general ways of defining the centre of a shape. In this case we see the masked area as the shape and its boundary as our shape boundary. One

type of centre is the actual middle: for both the x and y axis we find the minimum and maximum values, subtract the minimum from the maximum and divide by 2.

The other is the barycentre of the shape, also known as the geometric centre or centroid. It is defined as the average position of all points located within the shape, or the average position in the 2D grid of all points within the shape. In physics it would describe the centre of mass of an object.

The Euclidian distance between the middle and the centroid is a descriptor for the mask shape. If we for example have a circle or a square, the distance between the middle and the barycentre of the shape will be 0: they are one and the same coordinate. If however the shape of the mask is more asymmetric, but still within the same minimum and maximum values, the centroid will move in the direction of the shape with the greatest surface area. The middle will stay in exactly the same position and create a measurable distance. The normalised value of this distance is the final descriptor.

Although the centroid difference descriptor works well for convex shapes, it does not work well for concave shapes. Since a subset of the lesions we have in our datasets is convex, this feature can still be of use. For describing concave shapes, it would better to use descriptors based on the centre of the shape skeleton, or the point of maximal skeletal importance [48].

3.6.4 *Boundary Moments*

This descriptor was used by Kusumoputro in [24] and first described in this form by Gupta and Srinath in [49] and Shen et al. in [47]. In essence it uses functions based on the same statistical moment functions as used for the 2^{nd}, 3^{rd} and 4^{th} moments, known here as M_2, M_3 and M_4 respectively. But this time they are applied on the boundary of the lesion area instead of the whole skin lesion area.

We start with generating a path P of length N over all the pixels on the boundary. For each of these path elements we calculate, in order, the Euclidian distance between itself and the centroid of the lesion and store these in Z. If we would lay out these distances in order we get a graph that can be analysed. The 4 moments are defined by:

$$F1 = \frac{(M_2)^{1/2}}{m_1} = \frac{(\frac{1}{N}\sum_{i=1}^{N}(Z(i) - m_1))^{1/2}}{\frac{1}{N}\sum_{i=1}^{N} Z(i)}, \tag{25}$$

$$F2 = \frac{(M_3)^{1/3}}{m_1} = \frac{(\frac{1}{N}\sum_{i=1}^{N}(abs(Z(i) - m_1)))^{1/3}}{\frac{1}{N}\sum_{i=1}^{N} Z(i)}, \tag{26}$$

$$F3 = \frac{(M_4)^{1/4}}{m_1} = \frac{(\frac{1}{N}\sum_{i=1}^{N}(Z(i) - m_1))^{1/4}}{\frac{1}{N}\sum_{i=1}^{N} Z(i)}, \tag{27}$$

$$F4 = F3 - F1. \tag{28}$$

According to Shen, F4 was added because "... the difference between F3 and F1 can give information about the shape roughness which may not be represented by F3 or F1 individually." [47]. Kusumoputro adds to this that "The values of these moments are dimensionless, and invariant to transition, rotation and scaling. In general, the smoother the shape contour, the larger the value of the moments." [24]. The ranges of F1 to F4 are all positive and close to 0, therefore we use the range $[0, 1]$ for all four moments.

3.6.5 *Histogram of Boundary to Centroid Distances*

Besides determining the moments on the list of Euclidian distances Z we can also generate Z's histogram. Arques et al. make use of this feature in [50], although our specific implementation should be explained in more detail. Our version calculates the distance between the centroid and every pixel included on the boundary path P and not just on a subset of P. The histogram has the same number of bins as the other features' histograms (except HOG). To accomplish a good balance of the bins over the possible range of values, we spread out all the distances over the range $0 - (\frac{|P|}{4})$. Since the descriptor output is a histogram each attributes range is $[0, 1]$ and all histogram attributes together add up to 1.

The histogram's distribution gives a good description of the general shape of the lesion mask. If the lesion surface would form a perfect circle, centre-centroid difference would be 0 and all the distances between points on the border path and the centroid would be the same, resulting in a histogram where one bin is 1 and all the other bins are 0. If the border's shape is highly erratic (as in: there is a wide spread of distances in substantial amounts), all the bins of the histogram would be filled in more or less equal amounts. Although it must be said that such extreme cases are virtually non-existent. With most skin lesion masks we see that the middle five or six bins are filled.

3.6.6 *Asymmetry Index*

As the name of the feature implies, this descriptor represents the asymmetry of a lesion and was introduced by Parolin et al in [11]. It is implemented as follows: For every pixel p of p_i on the boundary of the lesion they calculate the Euclidian distance to the centroid c. This distance is compared with the distance between c and the pixel exactly opposite of p_i on the lesion perimeter: p_i'. This gives a list of distances d_i. The asymmetry index $AsymI$ is therefore defined as the smallest value of d_i.

Since it is relatively hard to implement the actual search for p_i', an alternative implementation is necessary. With the use of PCA we can determine two vectors, that together with their length, describe the two most prominent dimensions. As in: the two most principal components in the dataset containing all the boundary pixels. The asymmetry index is in this case defined as the difference in length between the two vectors, rescaled for the number of pixels in the image.

For the compactness descriptor, we have empirically determined that there will be no lesions with shapes that are very elongated. Therefore it seems correct to use a practical range of $[0, 100]$. This means that the measure will result in 0 if the two vectors of the most prominent dimensions have the same length. If the result is for instance 4, one of the vectors is 4 times as long as the other vector. However, this feature has the same disadvantage as the centroid difference descriptor in 3.6.3: it is only useful for describing convex shapes and not concave shapes.

3.7 FEATURE OVERVIEW

The following list gives an overview of all 33 features $\mathcal{F}$ included in the FGT, categorised by their class:

RGB descriptors

- Histogram
- Mean
- Standard deviation
- Variance
- Skewness
- Kurtosis
- Relative Chromaticity
- Colour Variance
- Local Binary Patterns
- Histogram of Sobel Edges
- G_x and G_y mean

RGB Co-occurrence matrix based texture descriptors.

- Homogeneity
- Correlation
- Contrast
- Uniformity
- Entropy
- Absolute Value
- Inverse Differences

Lab descriptors:

- Histogram
- Mean
- Standard deviation
- Variance
- Skewness
- Kurtosis
- Relative Chromaticity
- Colour Variance
- Local Binary Patterns

General texture based descriptors

- Histogram of Oriented Gradients

Boundary descriptors:

- Compactness
- Centroid Difference
- Boundary Moments
- Histogram of Boundary to Centroid Distances
- Asymmetry Index

3.8 EXCLUDED DESCRIPTORS

There are numerous other descriptors possibilities that could have been included in this project, but we had to draw a line somewhere. Several of these features are also quite complicated in their implementation and understanding. And since we were also interested in finding out how far we could come with relatively simple features (they are often better known and therefore more predictable), we decided to keep to this selection. We will however discuss several descriptors that have not been included:

- **Irregularity Index**: was included at first but later removed when it was discovered that it actually was another variant of compactness. It is defined in its mathematical form by:

$$Irreg = \frac{|P|^2}{4\pi|R|}, \tag{29}$$

  where $|P|$ denotes the length of the boundary path and $|R|$ denotes the area of the region of interest. The only difference with the compactness feature is the application of a 4π multiplier on $|R|$.

- **Fourier Feature**: as explained in Kusumoputro et al. [24]. In this descriptor, the shape of the lesion border can be described in complexity and roughness with a process involving the Fourier descriptor. It was not included for its complicated nature of implementation. Also: the tool already includes other lesion border descriptors (specially the histogram of of boundary to centroid distances) which describes the same details.

- **Granulometry Feature**: a more advanced and complex texture based feature. As explained by [39, 51, 52], this operator is used to calculate the distribution of particles in images of different sizes or shapes. Urbach et al. state that:

  > "Intuitively, a size granulometry can be considered as a set of sieves of different grades, each allowing details of certain size classes to pass. More formally, a size granulometry consists of an ordered set of operators, each of which converts an image to a new image in which features smaller than a particular size are absent." [52]

  Specific details within skin lesions of certain shapes and sizes are an identifiable property of said lesion. We decided not to include this feature since we want to focus on relatively simple features for this project.

- **Bag Of Visual Words**: a complex descriptor which stores image features in the form of (visual) words using a codebook generation process. It was first described in its current form using visual words instead of text words by Fei et al. in [53] and by Csurka et al. in [54]. For a good operation, the

image features need to be rotation, scaling and intention independent. Its large complexity was the main reason for excluding it.

- **Deep Features**: another complex feature and connected to the currently popular deep learning, which is a field within Machine Learning first mentioned by Hinton in [55]. It tries to replace the 'standard' human created features, like the ones we use in this thesis, with feature learning [56] and hierarchical feature extraction [57], using advanced versions of ANNs. The term 'deep' comes from using neural networks that have a relative high number of layers of neurons. Since deep learning only works well with datasets that contain a very large number of images (preferably hundreds of thousands to millions) and there simply does not exist such a dataset of pigmented skin lesion images, this group of descriptors was not included.

3.9 DISCUSSION

Implementation details on the FGT are given in appendix A.1. Computational performance is an important measure for application quality and usefulness in actual research and deployment, we have done speed tests with the `time` application on the single threaded `featuregen` tool. All tests were performed on a 2.3 GHz *Intel Haswell Mac* laptop with 16 GB RAM and an Solid State Drive (SSD).

Every feature of the list of features $\mathcal{F}$ used by FGT is $\mathcal{O}(cMN)$ for images of size M x N and constant c. If we have n images, this would mean that the complexity of FGT is $\mathcal{O}(ndMN)$ with constant $d > \sum_{i=1}^{\mathcal{F}} c$. We must however explain that since we have a substantial number of features, of which most have a $c > 1$, constant d will in most cases have a major influence on the running time of FGT. The size of the images and the number of images that have to be processed have in general the largest influence on the running time of the application.

When we look at how many images are processed in each dataset and the total time taken, we can conclude that the average time used per image is about 1.24 seconds, which can be described as sufficient for this amount of features per image. Note that all images have been reduced in size to fit within a window of 1024 x 1024 whilst maintaining there original aspect ratio.

Together with the .data output file, the tool also generated a .txt file containing a description in tree form of all the attributes it generates using a simple syntax. An example of a few lines of this file is shown in listing 1. This file can later be used to great effect in the FGT, as we will see in chapter 5.

The application is tested with the *Valgrind* memory debugging and memory leak detection tool under *Linux*, which concluded that there are no memory leaks or other problems. The tool is robust, but only when one does not deviate from a strict set of rules and images. The algorithm used to generate the path around the

mask boundaries cannot cope well with outliers or more then one mask on one image. Therefore we can enforce the following rule: every mask file can only contain a single connected component.

Listing 1: Top part of an attribute tree.

```
all;rgb;rgb_co_occur;lab;general;metadata
rgb;mean;std;hist;variance;skewness;kurtosis;sobel;rel_chrom;col_var;lbp
rgb_co_occur;vert;hor
lab;lab_mean;lab_std;lab_hist;lab_variance;lab_skewness;lab_kurtosis;lab_rel_
    chrom;lab_col_var;lab_lbp
general;centroid_diff;compactness;moments;pca_asymm_scale_index;border_path_
    hist;hog
metadata;name;height;width;mask_image_fraction
mean;mean_r;mean_g;mean_b
std;std_r;std_g;std_b
hist;hist_r;hist_g;hist_b
hist_r;hist_r_0;hist_r_1;hist_r_2;hist_r_3;hist_r_4;hist_r_5;hist_r_6;hist_r
    _7;hist_r_8;hist_r_9;hist_r_10;hist_r_11;hist_r_12;hist_r_13;hist_r_14;
    hist_r_15
...
```

4

CLASSIFICATION

Feature vectors alone do not result in classified skin lesion images. The classification problem still has to be solved (step 6 of figure 2). This classification problem has unknown datapoints that belong to one of two or more classes. A classification algorithms tries to determine to which class the unclassified datapoint belongs.

Classification problems can be divided into two groups: those that can be classified with linear classifiers and those that can only be classified with non-linear classifiers. A classification problem is linear when observations from two groups is separable by a hyperplane in a linear projection of the data. Such a hyperplane is also known as the separable boundary. A non-linear problem defines the exact opposite situation. In figure 10 we can see the difference between a linear and a non-linear problem. A non-linear classifier can be used to solve a linear problem, the opposite is not possible.

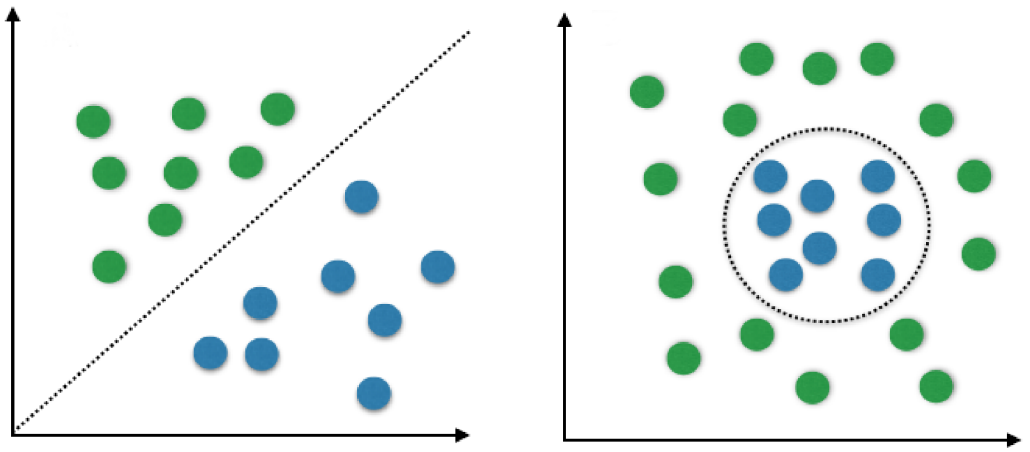


Figure 10: Subfigure (a) shows a linear classification problem. Subfigure (b) shows an example of a non-linear classification problem. Adapted versions of an image by Sebastian Raschka [58].

When we have more than two groups in our dataset we can still use linear classifiers, although in that case we would need more then one classifier. We could set up a structure where for every group we have a binary linear classifier with a corresponding test to check if it belongs to a specific group or not. If we combine

all classifiers with a decision (tree) structure we can use linear classifiers to classify more then two groups.

In chapter 2 we discussed that we had chosen two classifiers: kNN and SVM. First we will explain both classifiers in detail, followed by a discussion of the different normalisation modes of the tool. Thirdly we give an explanation of the tool made to apply the classifiers on our descriptor output data. We will end with a discussion section.

4.1 K NEAREST NEIGHBOURS

kNN is a simple and popular object non-linear classification algorithm with a history going back as far as 1951 [59, 60, 61]. It is a high quality classifier since it uses all available data and looks at multiple connections between the unclassified sample and other samples. It works by calculating distances between a new datapoint and all other datapoints in the dataset. The object is then assigned to the class which won the majority vote from a total number of k closest neighbours (see also figure 11). In general a k value $\geqslant 1$ is used since it prevents a new datapoint being classified as the class of an outlier. The k value is also often an odd number to get an automatic majority vote among the classes.

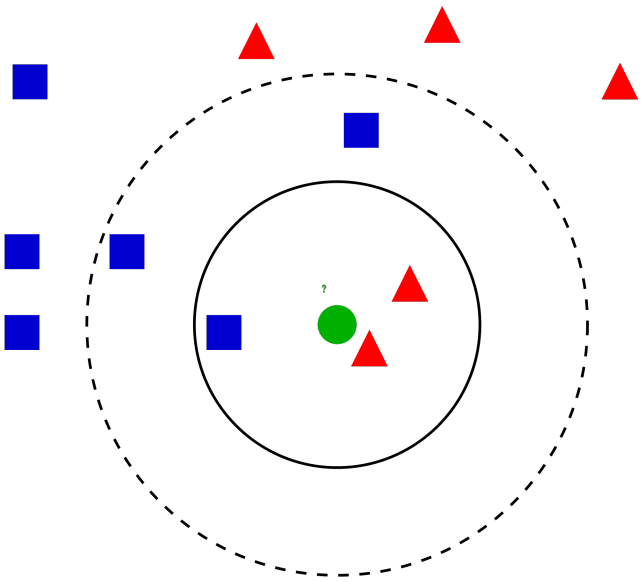


Figure 11: The green circle shaped datapoint needs to be classified. After distances are calculated between all blue squares and red triangles, a selection is made for either the $k = 3$ (inner closed circle) or $k = 5$ (outer dashed circle) closest datapoints. Image by Wikipedia user AnAj, public domain license.

kNN functions best in cases with large amounts of data and low noise levels. The need for large amounts of data can make kNN a slow classification algorithm. Performance of a standard kNN implementation takes $\mathcal{O}(n^2D)$ time, where D is the number of dimensions of the dataset and n is the number of observations in the dataset. In numerous cases where kNN is used, there is not enough data available. Therefore more efficient algorithms are needed.

Different distance measures can be used for determining distances between datapoints. In this project we use Euclidean distance. Although normally applied to LVQ, a number of alternative distance measures can also be used on kNN. A simpler measure is the Manhattan distance measure, which in some situations performs better (although tests for this project have concluded that performance in this situation is worse compared to using Euclidian).

A different group of distance measures can be found in asymmetric distance measures which in certain other cases can result in better classification performance then using Euclidian distance. For more information see [29, 30, 15] among others.

4.2 SUPPORT VECTOR MACHINES

SVMs grew out of the Vapnik-Chervonenkis theory, a theory developed between the 60's and the early 90's by Vladimir Vapnik and Alexey Chervonenkis in the Soviet Union as a major sub-branch of statistical learning theory. SVM in its current form was introduced by Cortes and Vapnik in [62].

Since SVM is a complex classifier, a solid explanation should be given. The following introduction into SVMs is based on that paper and a lecture by Patrick Winston for *MITs OpenCourseWare* [63] and by Russell and Norvig [31].

The basic SVM variant is a linear binary classifier. If it is possible to separate two groups with one separable boundary in a linear classifier, there are infinitely many options for such a boundary.

When describing methods it is convenient to use only two dimensions, but in practical situations there are often more dimensions present. For our explanation we can therefore imagine our hyperplanes as lines.

The basic goal of the SVM is to find the separating hyperplane that gives us the 'widest street' available between the two classes of data. The dashed line in figure 12 signifies the optimal hyperplane for our simple 2D example case and two closed lines defining our wide street. Each of the two lines are located on one or more datapoints from each class. These datapoints are called the support vectors. The distance between the two lines is defined as the margin. We want to maximize this distance to find the optimal margin. Therefore in its essence, SVM constitutes to margin maximisation.

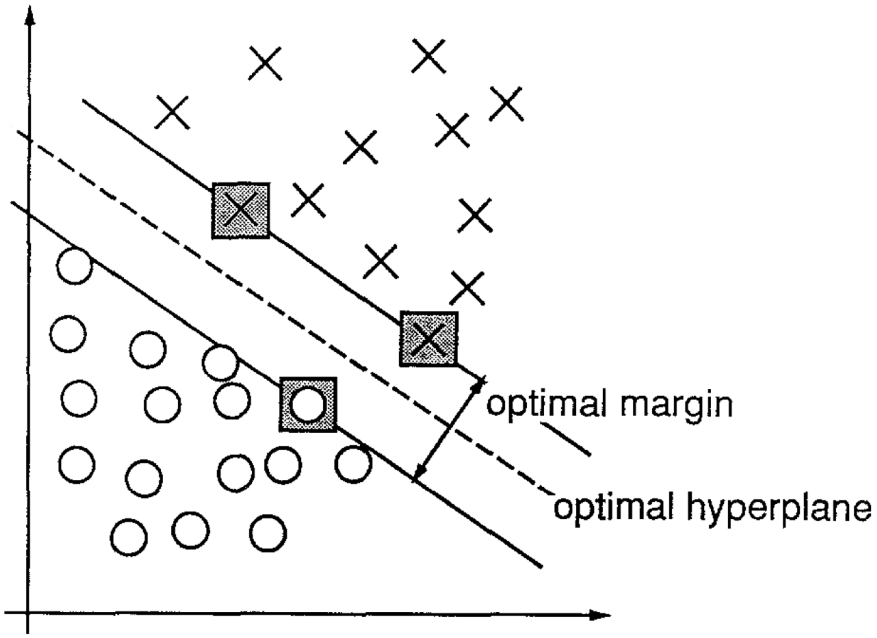


Figure 12: An example showing a 2D classification problem with two classes, one described with circles, the other with crosses. The grey squares show the support vectors that determine the margin of largest separation between the datapoints of the two classes. Image from [62].

4.2.1 *Definition*

To find this optimal hyperplane, we first need to define the decision rule that determines the hyperplane. With this decision rule we can also establish on which side of the hyperplane an 'unknown datapoint' (read: still to be classified) lays. The class on the right side (the crosses in figure 12) is defined as class $+$ and the class on the left side is defined as class $-$.

To accomplish this, a vector $\vec{w}$ is needed that is perpendicular to the median line of the street but with an at this moment unknown length. This makes $\vec{w}$ a normal of the separating hyperplane. The unknown datapoint constitutes a vector $\vec{u}$ with origin $(0, 0)$. To determine whether $\vec{u}$ lies on the left or the right side of the separating hyperplane, we take the dot product for a constant c: $\vec{w} \cdot \vec{u} \geqslant c$. We now have the distance proportional to the direction of $\vec{w}$ and using $b = -c$ we can rewrite our equation to:

$$\vec{w} \cdot \vec{u} + b = 0. \tag{30}$$

When the equation above is true, we know $\vec{u}$ to be on the right (as in not left) side of the separating hyperplane. Therefore, equation 30 is our decision rule. Now say we have a sample point $\vec{x}_+$ for which we know it belongs to class $+$. With equation

31 we want to insist that if a sample point belong to class $+$, the result should be equal or greater then 1. The same can be said for class $-$, hence:

$$\vec{w} \cdot \vec{x}_+ + b \geqslant 1, \tag{31}$$

$$\vec{w} \cdot \vec{x}_- + b \leqslant -1. \tag{32}$$

With a mathematical convenience we can combine both equations 31 and 32 by introducing an extra variable y_i, where y_i is $+1$ for samples from class $+$ and -1 for samples from class $-$. Hence equation:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geqslant 1. \tag{33}$$

For all the samples of $\vec{x}_i$ that are on the two hyperplane sides of the street, the equation can be simplified even further to form:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0. \tag{34}$$

Using this, we can now describe the distance between the two sides of the street, the length of the optimal margin, by taking the difference $(\vec{x}_+ - \vec{x}_-)$ between two samples that lay on opposite sides of the street. The distance can be calculated by turning $\vec{w}$ into a unit vector and taking the dot product of that with our difference to form:

$$(\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{||\vec{w}||}. \tag{35}$$

If we now enforce equation 34 onto equation 35, we come to the much simpler equation for our street width:

$$\frac{2}{||\vec{w}||}. \tag{36}$$

With our SVM we try to maximize equation 36. All this knowledge combined results in figure 13.

By dropping the constant and the division, we can state that we actually want to minimize $||\vec{w}||$. Or, for later mathematical convenience, we want to minimize $\frac{1}{2}||\vec{w}||^2$. For this optimisation problem we need to use Lagrange multipliers to form a new minimalisation function L. $\alpha$is a multiplier in this equation, which is described by:

$$L = \frac{1}{2}||\vec{w}||^2 - \sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]. \tag{37}$$

By finding the derivatives and setting them to 0, the minimalisation problem becomes much easier to solve. To do this, the derivative is changed to $\vec{w}$:

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \quad => \quad \vec{w} = \sum_i \alpha_i y_i \vec{x}_i. \tag{38}$$

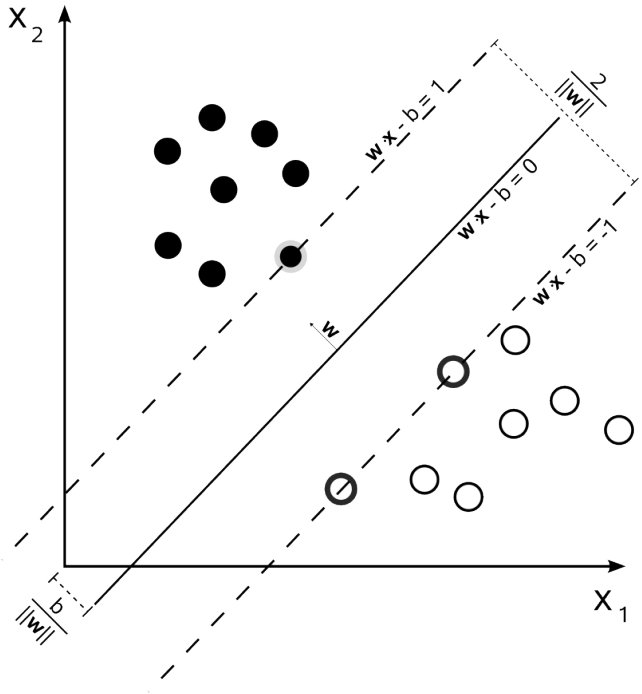


Figure 13: Detailing the maximum separating hyperplane and the margin. Image by Wikipedia user Peter Buch, public domain license.

Since b can also vary, its derivative has to be changed too:

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0 \quad => \quad \sum_i \alpha_i y_i = 0. \tag{39}$$

Now we can fill in equation 38 in equation 37, resulting in:

$$L = \frac{1}{2}(\sum_i \alpha_i y_i \vec{x}_i) \cdot (\sum_j \alpha_j y_j \vec{x}_j) - \sum_i \alpha_i y_i \vec{x}_i \cdot (\sum_j \alpha_j y_j \vec{x}_j) - \sum_i \alpha_i y_i b + \sum_i \alpha_i. \tag{40}$$

Since b is a constant, we can transform this long equation into a much simpler form. This new function needs to be maximised:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j. \tag{41}$$

Something important can now be concluded from L: our maximalisation problem is only dependant on the dot product of the samples $\vec{x}_i \cdot \vec{x}_j$. If this result is fed back into our decision rule with our unknown sample $\vec{u}$, we get:

$$\sum_i \alpha_i y_i \vec{x}_i \cdot \vec{u} + b \geqslant 0. \tag{42}$$

If this equation is equal to 'true', we can conclude that sample $\vec{u}$ belongs to class +. Therefore, the decision rule also only depends on one dot product: $\vec{x}_i \cdot \vec{u}$. As Winston points out [63]: it has been proven that the hyperplane optimisation problem that SVM solves after several iterations, is convex. It can therefore never get stuck in a local maximum.

4.2.2 *Kernels*

The default version of SVM is however only usable for linear problems. Using kernels, SVMs can be used in non-linear classification problems, making it much more powerful in many real world cases. The kernels enable the classifier to operate in a dimensionality that is higher than the datasets dimensionality using the mapping function $\phi(\vec{x})$. This higher dimensionality is also an implicit feature space.

We have shown earlier that we only need to maximise our dot products. Therefore it is now possible to wrap the mapping function around the two vectors of the dot product, giving us $\phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ and $\phi(\vec{x}_i) \cdot \phi(\vec{u})$.

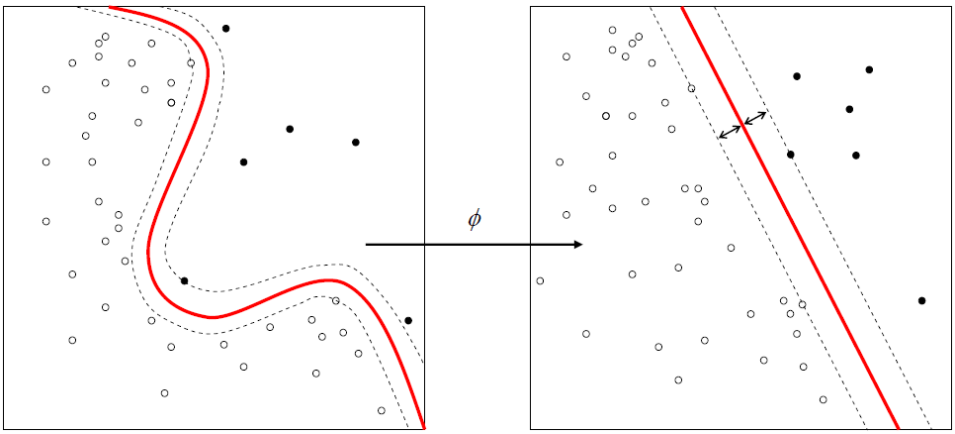


Figure 14: A kernel operation transforms a non-linear classification problem into a linear classification problem. Image by Wikipedia user Alisneaky, public domain license.

This results in a kernel function K as can be seen in:

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d. \tag{43}$$

This kernel function has the advantage that the original mapping function $\phi(\vec{x})$ does not have to be solved any more. This step is known as the kernel trick and was originally proposed by Aizerman et al. in 1964 [64]. The kernel trick provides the ability to increase the dimensionality without ever computing the coordinates of the data in that space. A simple visual depiction is given in figure 14. This basic kernel is known as the Linear kernel and is actually a version of the homogeneous kernel with a power d of 1. The inhomogeneous variant of this kernel is defined by:

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d. \tag{44}$$

There are many other kernels available besides the Linear, the most common being: Polynomials of higher degrees then 1, RBF or Gaussian and the Hyperbolic

Tangent. For the classification problem we have used two types: the Linear kernel (as seen in equations 43 and 44) and the RBF kernel as described by Vert et al. in [65], with σ being a free parameter:

$$K(\vec{x}_i, \vec{x}_j) = e^{-\frac{||\vec{x}_i - \vec{x}_j||}{2\sigma^2}}. \tag{45}$$

We name the Linear kernel version Support Vector Machine Linear (SVML) and the RBF kernel version Support Vector Machine Radial Basis Function (SVMR) from now on in this work. Compared to the Linear kernel, the RBF kernel can map the data to infinite dimensional spaces and can cope with multiple clusters of similarly classed samples.

4.3 NORMALISATION

Since classifiers compare values of the same descriptor type, a classifiers performance can be enhanced by applying a normalisation step. Feature vectors for every image are stored together in one dataset as rows, therefore all the elements of each descriptor are together in columns. The normalisation step is applied over all data (both the test set and the training set) in the dataset.

We have included two types of normalisation: scaling and standard score. With scaling normalisation, we take the minimum and maximum value of every column and rescale all the values in each column to be between 0 and 1. An alternative variant is the standard score, were for every value in a column the following is done: the column's mean is subtracted, after which the result is divided by the standard deviation of the column.

4.4 DISCUSSION

Implementation details on the *Classification Tool* are given in appendix A.2. Let us discuss its finer details and performance. First we will go over the speed of the classifier application with an example classification case on 753 images, where we have bin = 16, HOG bin = 16, no preprocessing, border features and a k value of 13 for when we use kNN. The results are shown in table 1.

From these time performance results we can conclude that SVMR is by far the slowest classifier of the three. The largest contribution to the slowness of the implementation is caused by the auto-train feature of *OpenCV* and the fact that the time the auto-train function takes is variable according to the dataset that needs to be classified.

Classifier	Time needed (s)
kNN	1.907
SVML	9.623
SVMR	116.829

Table 1: Speed performance of the three classifiers on our 'baseline' classification test of 753 images.

The classifier application is robust. If the input files and all the parameters are correct, the classifier tool has a predictable and stable outcome. Since the functionality and behaviour of the classifiers is known and stable, the accuracy performance generated by the tool is completely dependent on the dataset images, the descriptors that have been used and the k value for the kNN.

Andrew Ng gives an explanation on when to use SVML and when to use SVMR in [66]. According to him, it is best to use a linear kernel when the number of features is larger or equal to the number of observations. And SVMR is best to use when the number of features is about the same as the number of images up to the point where you have around 50 000 images. Since our number of features is in the hundreds and the number of images in most datasets is in the same range, it would make sense that the difference between accuracy scores between SVML and SVMR will not be big.

5

FEATURE SELECTION FOR CLASSIFIER CONSTRUCTION

Using the tools and systems we described in the previous chapters, we can now start with exploring datasets to reach our actual goal: a way to visually explore the feature space to determine a possible feature subspace. This feature space should have the following characteristics:

a) It should be smaller than the original full feature space.

b) It should lead to a higher quality classifier.

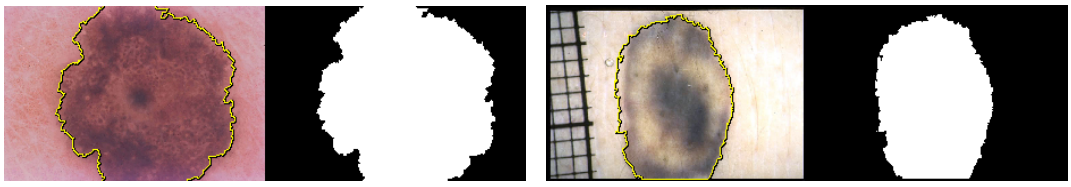
First we will give an overview of the datasets that are available to us. We continue with a detailed look at a feature exploration tool *Featured* that will be used to test the quality and usefulness of the descriptors. We will end with a section on the research plan.

5.1 DATASETS

Two datasets are used in this thesis. One is provided to us by the main supervisor, Dr. A. C. Telea and contains (for us) unclassified images. However, it is known that a subset of the images are melanoma. All images have been made with dermatoscopes. We will name this dataset *Dataset 1* and it contains 61 images, together with corresponding mask images. These masks have been made using a GVF based segmentation method made by Parolin et al. [11].

The other dataset comes from the *EDRA atlas dataset* [67] and has been provided to us by Dr. M. Emre Celebi of the Department of Computer Science at the Louisiana State University. We will name this set *Dataset 2* and it contains 1040 images made with dermatoscopes, together with a spreadsheet file containing metadata on all images. The images are classified into 20 classes, where each class contains at least a few dozen pictures. The spreadsheet contains a wealth of information about the categories used by physicians to classify the lesions.

In figures 15a and 15b we can see examples of images from *Dataset 2*. Here the original colour image and its mask are combined together, and the mask boundary has been highlighted in yellow on the original images. This yellow boundary is for visual observation by users, it is not included in the images used by the feature generation tool. The black size measurement mesh seen on the left side in figure 15b could have a slight influence on the outcome of the relative chromaticity and



(a) (b)

Figure 15: Two example images from *Dataset 2* with their corresponding masks.

colour variance descriptors, since those descriptors use colour values from areas surrounding the mask.

From *Dataset 2* we selected a total of 1021 images. Some of the images in this dataset do intersect the image border. We will focus on images that depict either naevi or melanoma, resulting in a selection of 753 images. 268 of these contain naevi and 485 contain melanoma.

All images and third party masks in the dataset were originally stored as .jpg or .png files but for our purpose they have been converted to .ppm and .pgm respectively. These lossless formats make it easier for our tools to work with them. Binary versions of .ppm and .pgm are used to save storage space.

5.2 FEATURED

The next major step in our pipeline is the exploration of the features we generated from the sets of images (step 4 in figure 2). To make this exploration possible, Paulo Rauber has made a Graphical User Interface (GUI) tool in Python called *Featured* (see figure 16). The paper "Interactive Selection of Image Features Aided by Dimensionality Reduction" [6], written by Paulo Rauber and among others co-authored by the author of this thesis, depicts the application and its general workflow in more detail.

The major idea behind this tool is to load a dataset of images and corresponding generated features, to plot a special 2D projection of all those feature and to be able to explore, both manually and with the help of scoring tools, the contribution of each of the features to the overall result. This way, a user can determine which set of features gives a high classification score.

The tool takes as input an .data file full of names of images and their corresponding feature data, as generated by either the FGT or from external sources. Together with the .data file it needs a folder with all the original images of the dataset and a feature tree structure .txt file.

Once the .data file and the images are loaded in, we can start to explore the dataset. In the top half of the screen we can see all the images of the dataset in the

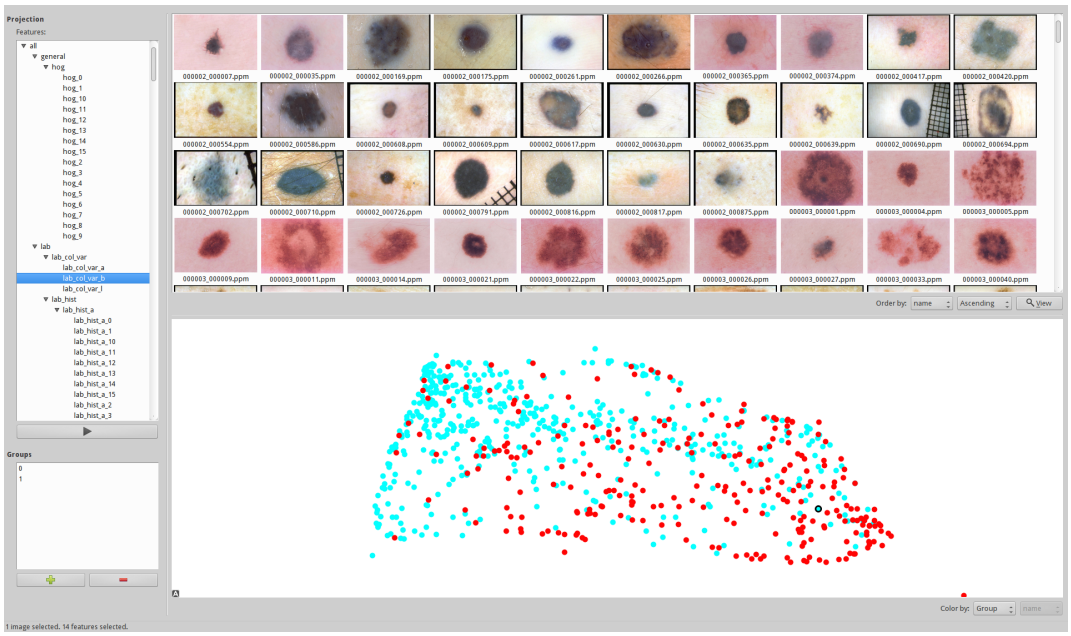


Figure 16: The application *Featured*, showing a subset of features from *Dataset 2*.

observation view. Images can be selected to see close-ups. The feature view is located on the top left side of the tool, showing a list of all the features of the datafile. These are generated from the feature tree structure file together with all the correct feature labels as described in chapter 3. The lower left side of the tool contains the group view, showing the different groups the dataset is divided into, either from the original data, or from selections made by the user.

The projection view in the lower window shows a 2D scatter plot of the projection of the featurespace generated using one of several dimensionality reduction methods. The datapoints in the scatter plot can be coloured according to user-selected features from the feature view or user-defined groups from the group view.

The most important projection technique is the Least Squares Projection (LSP) method, as implemented by Paulovich et al. [68]. LSP uses least squares approximations to find the coordinates of the points in the projection from the coordinates of a subset of the points of the dataset, the control points. An important ability of the LSP technique is that it tries to preserve similarity relationships between neighbouring datapoints as much as possible.

When a projection of a dataset is made, a user can explore its scatter plot using several selection options. A subset of images in the plot can be selected with the standard mouse-dragging operation. The corresponding images in the observation view are then highlighted. Selecting a subset of features in the feature view immediately generates a new projection.

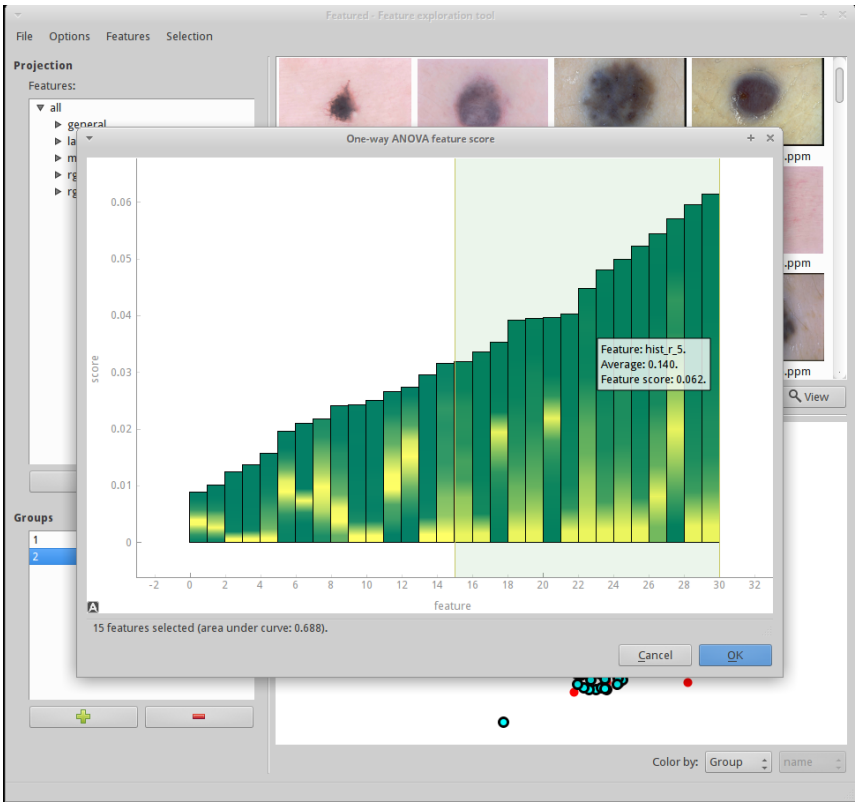


Figure 17: *Featured* showing a table of scores generated with a one-way Analysis of Variance (ANOVA) test in the feature scoring view. Sliding a threshold line from left to right enables the user to select features.

The tool can generate scoring sheets of all the features based on several relevance metrics from three groups: univariate, multivariate and recursive feature elimination. These scores are shown in the form of a bar chart, from least relevant feature on the left side to most relevant feature on the right side, in the feature scoring view. An example of this view is shown in figure 17. The user can select a subset of features, from which a new projection is generated and shown in the projection view.

When either all or a subset of features is selected (step 5 in figure 2), a .data output file can be generated for use in classification. More details about the tool can be found in the aforementioned paper [6].

5.3 FEATURE SELECTION WORKFLOW

From our main research question from chapter 1 and all the methods and tools we just discussed, we can now plan our research. We have shown some effective algorithmic building blocks and generated tools with them. Since we have already answered our first sub-question in chapters 3 and 4, and our second sub-question

in this chapter, we can now focus on the main research question: How can we empower the designer of skin lesion classification tools to effectively and efficiently explore the design space of skin classification algorithms?

With the combination of the FGT, *Featured* and the *Classification tool*, it is possible for a classification system designer to explore the descriptor space and determine which set of features gives the highest classification accuracy. We first need to test all the other parameters and options of the FGT and the classifiers to determine how we are going to achieve high accuracy scores, resulting in the following four main cases:

1. Define our baseline parameter combination and generate classification result on baseline case.

2. Generate all variations of parameters on baseline and generate classification results from them.

3. Test different feature subset types.

4. Test unknown images from a different test dataset.

This baseline case is created to compare parameter options against a reasonable default set of settings. The number of combinations of all parameters and options is large in such a degree that it is not feasible to generate them all in this project. Therefore it is best to define a baseline case and then to test all the options compared to it. We can now start testing and comparing different subsets of images with the baseline case, hopefully resulting in a score that is as high or higher than our baseline case.

Our baseline case includes the following: 753 naevus and melanoma images from *Dataset 2*, 16 histogram bins, 16 HOG bins, no boundary features, no pre-processing and standard score normalisation. 5-fold cross validation is used to generate our classification results.

With k-fold cross validation, the original set of samples in the dataset is partitioned into k equal sized subsets. Therefore with 5-fold cross validation, a k value of 5 is used. Of the k subsets, one subset is held back as validation data for testing the model, and the remaining $k-1$ subsets are used as training data. The cross-validation process is repeated k times, known as the folds, with each of the k subsets used only once. The k results from the folds can then be averaged to produce a single score: in our case the classification accuracy. The advantage of this method is that all samples in the dataset are used for both training and testing and that each sample is only used once.

Combining all factors mentioned above results in 10 main test cases:

1. Baseline case test to determine the optimum k value for kNN.
2. Baseline case for the 3 classifiers: kNN, SVML, SVMR.
3. Baseline case normalisation test.
4. Baseline case preprocessing test.
5. Number of histogram bins.
6. Border features.
7. Influence of descriptors classes.
8. Subsets of features.
9. Best of all.
10. Compare *Dataset 1* containing non-previously-classified images with the baseline training set on *Dataset 2*.

The results for these 10 tests will be given in the 6 chapter and they subsequently will be discussed in chapter 7.

6 RESULTS

This chapter will show and explain the results of the 10 test cases shown at the end of chapter 5. Tables, plots and images of projections are included to better illustrate the results. A discussion of the results will follow in chapter 7.

TEST 1: BASELINE K VALUE TEST

The kNN classifier applied to the baseline case with *Dataset 1* gives the results shown in figure 18. From these results we can conclude that $k = 13$ corresponds to the best kNN classification accuracy score of 0.769. From now on we will use $k = 13$ for kNN. However, the differences in classification score between $k = 5$ and $k = 25$ seem minor.

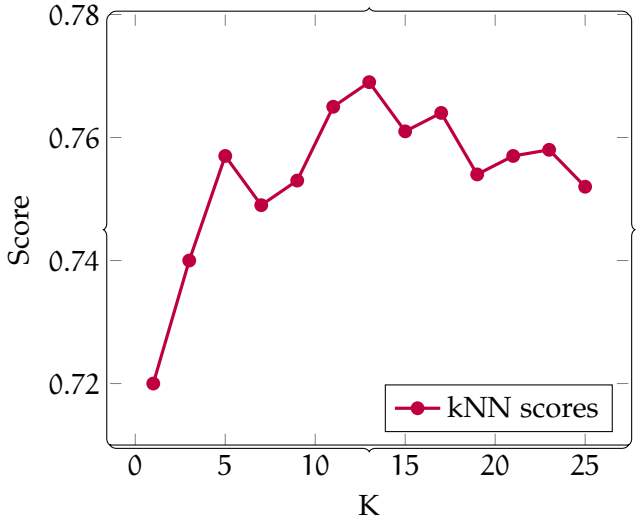


Figure 18: kNN scores of baseline case.

TEST 2: BASELINE CLASSIFIERS TEST

Now that we have chosen our default k value, we can compare the classification scores of SVML and SVMR with kNN. The accuracies are listed in table 2. Here we can see that we have a 4% spread in accuracy score between SVML and SVMR, with kNNs performance right in between them.

Classifier	Score
kNN	0.769
SVML	0.748
SVMR	0.788

Table 2: Comparing kNN, SVML and SVMR classification accuracy for baseline case.

TEST 3: BASELINE NORMALISATION TEST

For our third test we will compare the two normalisation methods for our three classifiers. On the bottom axis we have our 3 normalisation modes: 0 for no normalisation, 1 for scaling and mode 2 for standard score. If we look at table 3 and the corresponding figure 19, we can see that not applying normalisation always has a negative effect on the accuracy of an classifier. We can conclude that scaling normalisation results in the highest classification score for SVML, that kNN favours standard score normalisation and that SVMR favours both versions of normalisation over no-normalisation.

	Norm		
Classifier	0	1	2
kNN	0.689	0.737	0.769
SVML	0.506	0.787	0.748
SVMR	0.700	0.789	0.788

Table 3: Differences in classification score for the different normalisation modes.

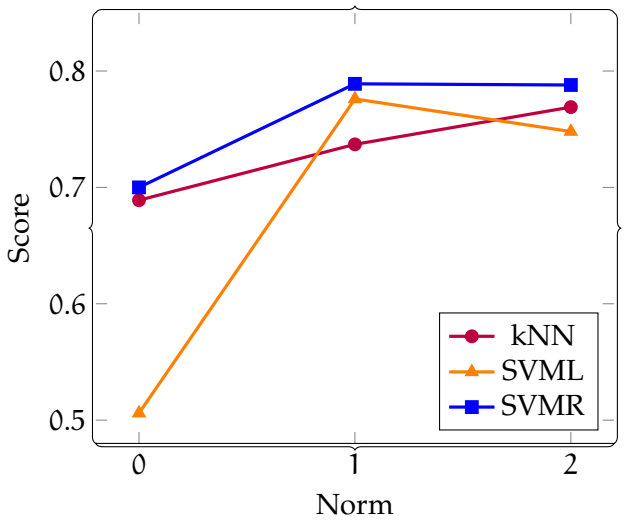


Figure 19: Differences in classification score for the different normalisation modes, based on table 3.

TEST 4: BASELINE PREPROCESSING TEST

To test the pre-processing options for our classifiers, the three modes that were explained in chapter 3 are compared with no pre-processing. Mode 0 is no pre-processing, mode 1 is unsharp mask filtering, mode 2 constitutes contrast stretching and in mode 3 contrast stretching is followed by applying the unsharp mask filter.

When we look at table 4 and figure 20, we can see that applying contrast stretching has a positive effect on the accuracy score for kNN and SVMR. Mode 3 also gives a slightly higher score for SVMR compared to mode 0, but the difference is too minimal for stating that there is a definite improvement. SVML is however behaving unexpectedly. Both modes 1 and 2 have a substantial negative effect, but applying mode 3 has a slight positive effect. This is also unexpected behaviour. Both mode 1 and 2 have a very big negative influence on the classification score and should not be applied. However, mode 3 does seem to have a slight positive effect on the score.

	Pre-processing mode			
Classifier	0	1	2	3
kNN	0.769	0.761	0.778	0.762
SVML	0.748	0.691	0.697	0.764
SVMR	0.788	0.784	0.805	0.793

Table 4: Differences in classification score for the different pre-processing modes.

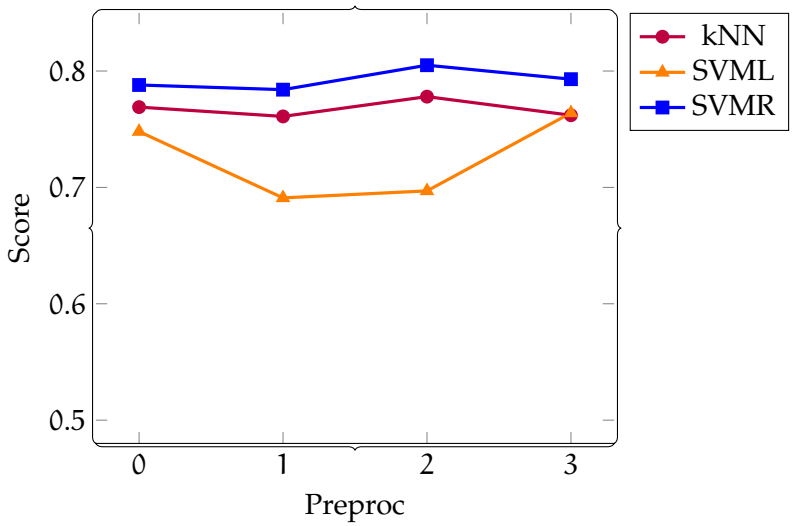


Figure 20: Differences in classification score for the different pre-processing modes. Corresponding to table 4.

TEST 5: NUMBER OF HISTOGRAM BINS

The number of bins for the general histograms and HOG descriptor histogram of the baseline case are both set to 16. For more information on this, we refer back to section 3 on histograms in the FGT. This test is an attempt to determine if deviating from these default values has a positive or negative effect on the classification accuracy. In both cases we have calculated all the baseline features for four example values for the number of bins: 8, 16, 32 and 64 (all out of a possible 256).

	Number of bins			
Classifier	8	16	32	64
kNN	0.768	0.769	0.758	0.748
SVML	0.718	0.748	0.728	0.730
SVMR	0.809	0.788	0.789	0.786

Table 5: Differences in classification score for the different histogram bin sizes.

First up is the standard histogram bin size. Table 5 shows that, although the differences are not large, there are differences of 1.5% to 3% between the lowest and the highest accuracy score for each classifier type. kNN achieves higher accuracy scores when 8 to 16 bins are used. A number of 16 seems to be great for SVML. SVMR prefers a lower number of bins.

The results in table 6 shows that SVMR operates along an different pattern for the number of bins used by the HOG descriptor. It prefers a larger number of bins used

	HOG number of bins			
Classifier	8	16	32	64
kNN	0.771	0.769	0.773	0.769
SVML	0.732	0.748	0.730	0.716
SVMR	0.787	0.788	0.791	0.793

Table 6: Differences in classification score for the different HOG bin sizes.

by the HOG descriptor. kNN accuracy scores are overall unresponsive to changes in number of HOG bins. Finally SVML performs best with the default number of 16 bins.

TEST 6: BORDER FEATURES

To test the influence of the border features we have applied the classifiers on the baseline set of features on the 753 image dataset from *Dataset 2*, in combination with the border features. In table 7 and the corresponding figure 21 we can see the results for the three classifiers.

	Norm		
Classifier	0	1	2
kNN	0.690	0.733	0.756
SVML	0.576	0.779	0.701
SVMR	0.694	0.798	0.782

Table 7: Differences in classification score for the different normalisation modes when using border features.

If we compare these results with the classification results in table 3 from test 3, we can conclude that the overall pattern and shape of the graphs with and without border features is not identical. There is not a significant increase in accuracy score when the border features are used. For example: SVMR combined with scaling normalisation without border features gains a score of 0.789, but with border features that score slightly increases to 0.798.

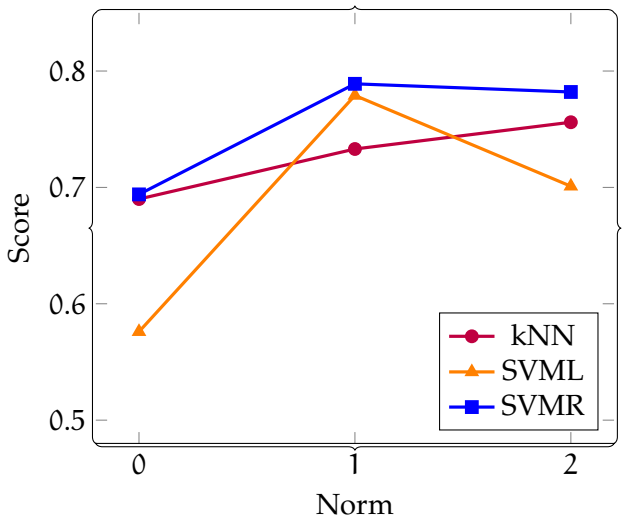
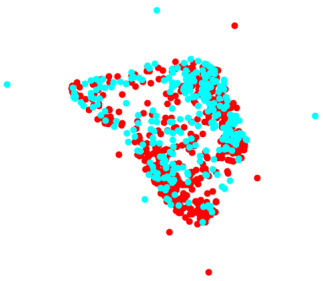


Figure 21: Classifier accuracy scores when border features are enabled for different normalisation modes. Corresponds to the results in table 7.

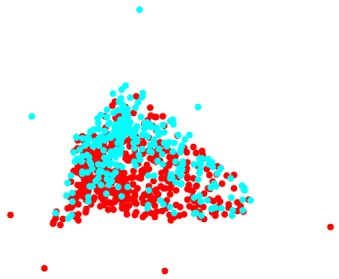
TEST 7: INFLUENCE OF DESCRIPTOR CLASSES

Before testing subsets of features, it is recommended to have a look at the spread and effect of the descriptors. Image 22a shows the projection of all the descriptors, feature vector of length 369, generated by the FGT except from the 4 metadata features. In images 22b, 22c, 22d, 22e and 22f we can see the projections made for the 4 different major groups of descriptors and the HOG feature. All the projection images have been generated by *Featured*. All the red datapoints in the projections belong to the naevus class images and all the blue datapoints belong to the melanoma class images.

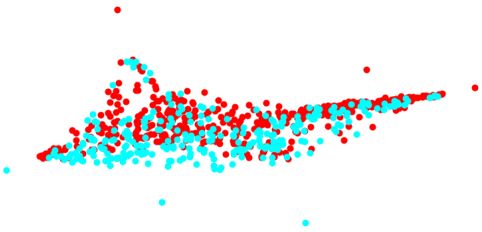
Some of these projections have peculiar shapes, especially the one shown in figure 22e. The more important observation however, is the fact that there is no clear separation between the two classes in any of these projections. There is always a major overlap in projection areas of both classes. Even more important: the projection of all features in figure 22a is not significantly different in overlapping area compared to the other major projections. From this we can conclude that using just one of the major descriptor classes does not guarantee an above average classification score.



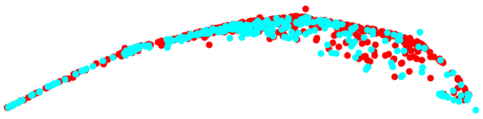
(a) Projection of all features.



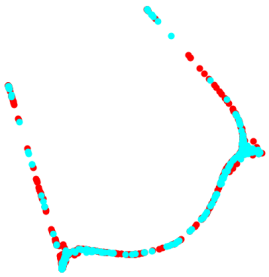
(b) Projection of all RGB features.



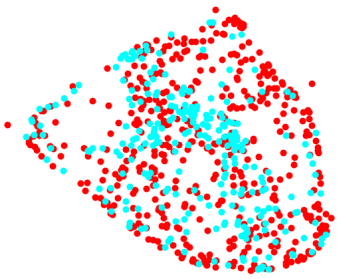
(c) Projection of all Lab features.



(d) Projection of RGB co-occurance matrix features.



(e) Projection of all HOG features.



(f) Projection of all border features.

Figure 22: (a)-(f): all are LSP projections showing the distribution of datapoints for several descriptor class combinations.

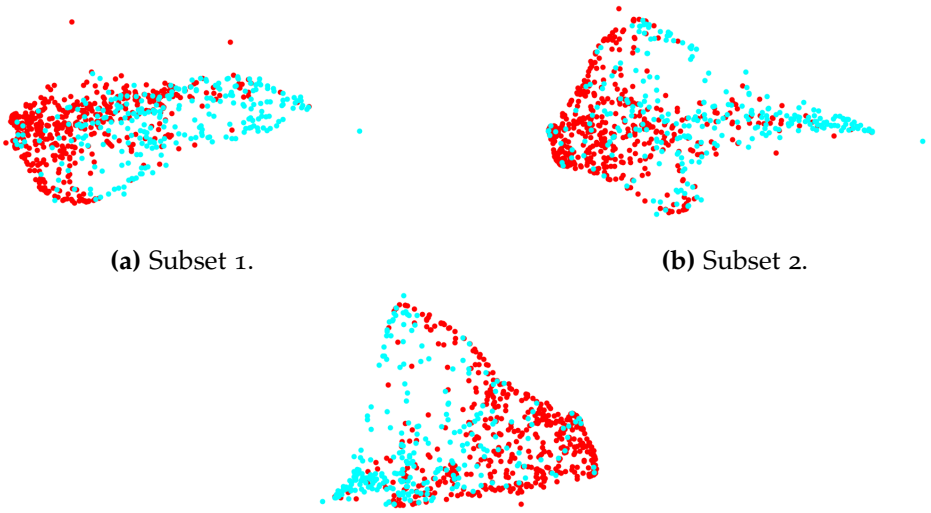
TEST 8: SUBSETS OF FEATURES

Using *Featured* to explore the feature space, we have come up with three subsets. Firstly, we have subset 1, containing a feature vector of length 16:

Subset 1

- hist_r_3
- hist_r_4
- hist_r_5
- hist_r_7
- hist_g_5
- sobel_hist_g_5
- col_var_r
- col_var_b
- lab_std_b
- lab_hist_l_1
- lab_hist_l_2
- lab_hist_l_5
- lab_hist_b_7
- lab_hist_b_8
- lab_hist_b_9
- lab_variance_b

For example: 'hist_r_3' stands for the third bin of the histogram of the red colour channel. A LSP of these descriptors can be seen in figure 23a. Here we can see a slightly larger group of blue melanoma datapoints that appear to be slightly more separated from the other mixed datapoints compared to the previous projections in figures 22a to 22f. This seems to point at a better separation existing between the two classes. But there are still many blue melanoma datapoints spread out in between the red naevi datapoints. Therefore we cannot make any definitive conclusions yet.



(a) Subset 1.

(b) Subset 2.

(c) Subset 3.

Figure 23: LSPs of the three subsets.

Next up is subset 2, which was created by applying the randomised decision tree feature scoring method in *Featured* to the naevus class and manually selecting the top few features from an empirically chosen threshold. Its projection can be seen in figure 23b.

Subset 2

- hist_r_2
- hist_r_3
- hist_r_4
- hist_r_5
- hist_r_6
- lab_std_b
- lab_hist_l_2
- lab_hist_l_4
- lab_hist_b_8
- lab_hist_b_9
- lab_variance_b

Finally we have subset 3, created in the same manner, but now applied to the melanoma class. Its projection can be seen in figure 23c.

Subset 3

- hist_r_3
- hist_r_4
- hist_r_5
- hist_r_6
- lab_std_b
- lab_hist_b_8
- lab_hist_b_9
- lab_variance_b

As we can see, there are many descriptors the subsets have in common. Even more specifically, subset 3 is a subset of subset 2, which in turn is almost a subset of subset 1.

	Set of features			
Classifier	Baseline	Subset 1	Subset 2	Subset 3
kNN	0.769	0.770	0.773	0.772
SVML	0.748	0.782	0.781	0.777
SVMR	0.788	0.786	0.778	0.765

Table 8: Differences in classification score between the baseline version, subset 1, subset 2 and subset 3.

We compared the classification results from the three classifiers with our baseline case in table 8. kNN performs slightly better with any of the descriptor subsets, but all differences are within 0.5% of each other. SVML performs substantially better with any of the subsets, attaining its highest score with subset 1. For SVMR we can conclude that it performs as well with subset 1 as with all the features. However, subset 2 and especially subset 3 show a significant lower performance. We have attempted to find other interesting subsets but they all had substantially lower accuracy score and were therefore deemed not relevant enough to be included in this thesis.

TEST 9: BEST OF ALL

All the previous tests combined with the parameters for which each option in the tool attained the highest accuracy scores, should result in an overall 'maximum' classification score. We have picked two parameter combinations to test this. For best set 1 we run the FGT and Classifier Tool with the following settings: SVMR classifier, scaling normalisation, contrast stretching pre-processing, 8 histogram bins and 64 HOG histogram bins while also using all features including border features.

For best set 2 we run with the following settings: SVMR classifier, but now with standard score normalisation, contrast stretching pre-processing, 8 histogram bins and 64 HOG histogram bins and again using all features including border features. The total feature vector length for both these versions is 289. The results can be seen in table 9.

	Set of features		
Classifier	Baseline	Best set 1	Best set 2
SVMR	0.788	0.806	0.822

Table 9: Classification score between the baseline version and the two best sets.

Selecting the correct combination of parameters seems to have a significant influence on the accuracy score. From all tests we have done, the combination best set 2 generates our highest accuracy score of 0.822.

TEST 10: TEST SET *Dataset 1* COMBINED WITH TRAINING SET *Dataset 2*

For our final test we compare non-classified images from the previously mentioned *Dataset 1* with *Dataset 2* as the training set. Baseline parameters are used for all datasets and SVMR is used as the classifier. The correct classification of these two

sets, as previously created by medical specialists for one of the supervisors, was not known to the author of this work at the time of testing.

When we run this setup for the 61 images from *Dataset 1*, all images are classified as belonging to the benign naevus group, except for one image, which was classified as being a melanoma. Two dermatologists have rated this same dataset and stated that of the 61 images, they would classify two as being 'borderline' between benign and malignant, while all the other images would be classified as benign.

7

DISCUSSION

Did we actually solve our initial problem? And if so, to what extent? In this chapter we will go over the observations and conclusions that can be made based on all previous work for this project. Specific improvement propositions will be explained in detail in chapter 8 and our final conclusion will follow in chapter 9.

7.1 SEGMENTATION AND PRE-PROCESSING

Since this thesis does not attempt to give a qualitative comparison of segmentation methods applied on skin lesion images, we have no conclusions concerning this available. However, we can state that both the GVF and superpixel methods give potentially high quality segmentations of skin lesions. Since both methods have a manually controlled first phase, their final segmentation quality depends significantly on human influence. The test results of chapter 6 do show that the segmentations used in this thesis are good enough for high classification results.

The pre-processing test in test 4 of chapter 6 shows that pre-processing in general has the potential to improve accuracy scores: contrast stretching helps both kNN and SVMR achieve better classification results. As stated in chapter 3, the three modes of pre-processing tested in this work might not be the best ones available. Therefore any final conclusion cannot be made yet. However, as seen in test 9, including contrast stretching in the best combination of parameters does suggest that it has a positive effect in making the accuracy score substantially higher.

7.2 DATASET QUALITY

The quality of the dataset has a major influence on the outcome of the classification process. The images should be taken with the camera being straight above the lesion. Lighting should be consistent and without shadows. White balance (also known as the colour temperature) should be neutral. The images should have been segmented correctly and there should not be too much healthy skin included within the skin lesion section.

However, the resolution of the image does not have to be exceptionally high since it would only slow down the feature extraction process significantly. In addition,

the quality of the extracted features does not advance from when a certain level has been reached. Images should also not be too small in resolution, since in that case we cannot extract all required information.

Another factor of training set quality is the aspect of dataset symmetry. It is best for the performance of an classifier if a dataset with $\geqslant 2$ different classes has an equal number of elements (in our case images) in each class. If the number of elements per class is equal, the dataset is balanced or symmetrical. If the number is not equal, the dataset is unbalanced or asymmetrical.

A training set which has a disproportionately high number of elements of one class will result in a classifier that is biased towards this majority class. When a classification algorithm that has been trained on such data is applied to a test dataset that is also unbalanced, the classifier will give results which are too optimistic.

The training set used in this project is asymmetric: one class contains 268 images, the other contains 485 images. However, we have seen datasets used in affiliated literature that are even more unbalanced. This unbalance does not mean that the accuracy scores achieved in with our research are invalid, but the knowledge that the dataset is unbalanced does have to be taken into account by the reader.

Certain datasets used in research done by others contain both images made with standard digital cameras technology and images made with dermatoscopy methods. As explained before in chapter 2, these two types of images are really different from each other. It therefore seems unwise to use both types together in one dataset. The images in the datasets used in our research have all been made with dermatoscopes.

The amount of effort needed to create a dataset that is consistent in its image capturing quality, large enough (> 1000 per skin lesion class for instance) and well balanced between skin lesion classes and the fact that such datasets are not readily available yet implies that there is still a major problem present.

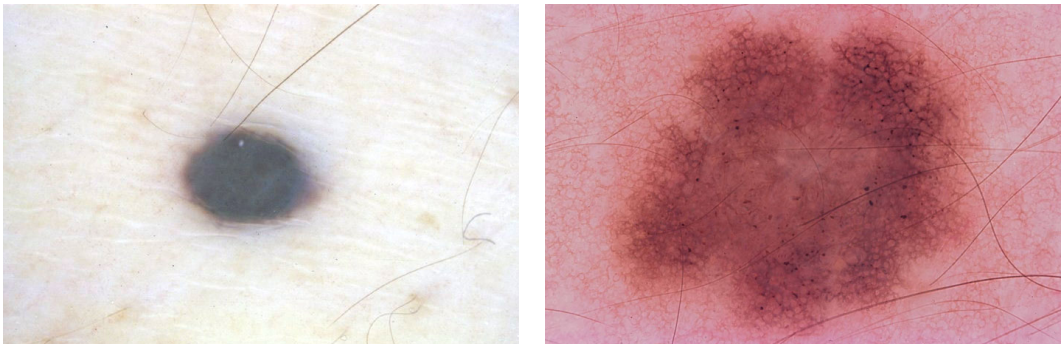
This is probably caused by the fact that a substantial amount of time is needed by medical professionals who are already short on time in general, together with the strict ethics and privacy rules regarding medical information of patients currently present. Even though it is good that these rules exist, the fact that all research in skin lesion classification can be performed on anonymous data suggests that our problem should not be as substantial as it currently is.

Especially the number of images in the dataset is of great importance. The datasets many others have used are too small. A low point in this case has been achieved by the set used by Kusumoputro and Ariyanto in [24], which had only 29 images with malignant melanoma and 34 images containing benign lesions.

7.3 OTHER DATASET QUALITY CONSIDERATIONS

Darker human skin colours have generally been avoided so far in most existing literature for the fact that it is considerably more difficult to apply a feature based classification system in such occasions. The contrast and colour differences between healthy and unhealthy skin are much smaller compared to lighter skin colours. Likewise, there is hardly any dataset available containing images of people with darker skin colours.

It would be safe to state that research within this field requires datasets with a higher variety in skin colour. To accomplish this it would be recommended to adjust the privacy rules to enable required data to be published and used. Since there is no way for the skin lesion classification system researcher to link an skin lesion image to the person it was taken from there should be no privacy issues.



(a) **(b)**

Figure 24: Differences in general skin colour present in datasets. Image (a) shows an example where the human skin seems almost too white. Image (b) shows an example where the human skin seems unrealistically red.

But if we compare figures 24a and 24b with each other, it is clear where our problems lie: there can be a great variety in human skin colours or lighting situations causing problems for automatic classification systems. In this case we will have a problem with the reddish colour of the 'healthy skin' in figure 24b.

But why is the skin red in this case? Does the patient have a rash? Or did he or she exercise just before the image was taken? Or are the camera settings incorrect, causing a general red hue, which also affects the colour of the skin lesion in the image? We can at least conclude that the tone levels of the images in the dataset must be consistent.

In specific cases such as when a patient has a rash around the skin lesion, we cannot assess the average skin colour. Maybe a larger skin area of the patient must be photographed whereby a piece of the patients actual healthy skin becomes visible.

7.4 FEATURE EXTRACTION AND SELECTION

Test 5 showed that the number of bins used in histograms can have a small but still significant influence on the scores. However, since not all numbers have been tested, the optimal number of bins cannot be given. But if more tests are done, it should be possible to finalise which amount of bins give the best result.

Even though border features are of major importance for classifying skin lesion by humans, the classification scores of test 6 show that in our case we have some mixed results. On its own, the addition of border features only increases the classification score in a few cases, most notably with the use of SVMR in combination with scaling normalisation.

As shown before though in the sources mentioned in chapter 3, more border features in general seem to have a substantial contribution in automatic classification systems. Using border features also does not negatively affect the score of the two best sets in test 9 compared to the baseline case. Both observations contribute to our conclusion that the border features are important to an automatic skin lesion classification system.

As stated in test 7 of the results chapter, there is no clear separation between the two classes of naevus and melanoma in any of these projections. The subsets of test 8 however show that certain subsets of descriptors can result in accuracy scores that are as good, or sometimes even better, than the original scores with all descriptors included.

However, in the end none of the tested subsets performed better then the SVMR classifiers using all features. Therefore we conclude that if there is a need in some future version of an automatic skin lesion classification system to enhance the performance of the system by reducing the number of descriptors, it is possible to use a subset of descriptors. This option should therefore only be used if the baseline classification score including all descriptors is already high. Basically: a score that is significantly higher that our current classification scores.

7.5 CLASSIFICATION

We can now derive several conclusions based on the tests in the previous chapter. The results of the three classifiers kNN, SVML and SVMR are close to each other if their respective optimal parameters are used. This means there are other factors that have a larger influence on the classification accuracy than just the choice of classifier. These factors include: quality of dataset, the descriptors used, preprocessing steps and the segmentation quality. However, this is only true for proven high quality classifiers. Any of the other classifiers mentioned in chapter 2 should be able to achieve comparable results.

We can conclude from test 3 that applying normalisation has a positive influence on classification accuracy in general. kNN prefers standard score normalisation, SVML prefers scaling normalisation and SVMR works well with either. Our best classification accuracy score of 0.822 in test 9 shows that selecting the correct classifier, pre-processing step, number of bins and normalisation mode have a significant influence on the accuracy score.

From the results of test 10 we can conclude that for the tested dataset *Dataset 1*, which is of a completely different origin than the training dataset, our approach labelled images almost completely correctly in comparison to the observations made by two dermatologists.

7.6 COMPARISON WITH LITERATURE

Several classification scores achieved by others in the field are given in chapter 2, but will be repeated here. Are our accuracy scores comparable with respect to the compatibility of our scores? Since almost all datasets used in related works are different, an exact comparison is not possible. Cheng achieved "... an overall classification success of 79%, with 70% of the benign lesions successfully classified, and 86% of malignant melanomas successfully classified." [4] using a multi-layer perceptron neural network model.

Cheng's overall accuracy score of 0.79 [4] is slightly lower then our maximum score of 0.822, but we can conclude that our scores are in the same ballpark. Parolin et al. in [11] attained overall accuracy scores between 0.8255 and 0.8841, depending on how low their false negative rate was set. Which means that our score is only between 0.5% and 6.2% lower than their accuracy score. Their score is therefore reasonably similar to ours.

Celebi et al. in [13] achieve an accuracy score that is somewhere between 0.92 and 0.93, and therefore substantially higher than our best score. Their dataset contains about the same number of images as our set, 564 images, but with only 88 of those are images containing malignant melanoma. Their image set is significantly more unbalanced in comparison with the set used in this work, which makes comparing it with our accuracy scores less relevant. We can conclude that our accuracy scores are consistent with results attained by others in the field.

8

FUTURE WORK

From the previous discussion chapter 7 we can conclude that there are still ample opportunities for improvements. We will therefore give a list of these opportunities and their details in this chapter.

- **Segmentation Process**: Even though it was determined earlier that the superpixel segmentation method in many cases results in the highest quality semi-autonomous segmentations currently possible, in some cases the current segmentations are not good enough.

  However, certain questions still remain. When a segmentation method creates a boundary based on the lesion, does this boundary follow the actual lesion's shape in close enough detail? Certain methods might create either a more jagged or a smoother boundary than the actual lesion actually has, which has a major influence on the credibility of the boundary features as actual useful descriptors.

  Since seed points have to be placed manually, what is the exact influence of the seed placement to the correctness of the resulting mask? Is there actually a way to automatically place these seeds? Or is this theoretically not desirable in the first place? These questions should be answered by doing more research into ways of achieving even higher and more stable segmentations the classifiers can use.

- **Pre-processing**: From the section on pre-processing in chapter 3, the results in chapter 6 and the discussion in the previous chapter we concluded that the performance of the pre-processing options tested in this work was not sufficient enough to warrant their use. In literature however, other options for this phase in the pipeline have been mentioned that could have a larger positive impact on accuracy performance. A more in-depth test on pre-processing alternatives should therefore be conducted.

- **Descriptors**: As explained in the discussion section of chapter 3, there are many other options for descriptors not yet used in this work. Descriptors such as: dermal features if metadata is present, granulometry features, Fourier based features, LBP with local areas, bag of visual words, features using pattern spectra based on connected filters in colour or even wavelet based descriptors. There are also many more methods available to describe textures. What would their influence be?

- **Classifiers**: Since only two main classifier groups were tested in this thesis, it would be better if several other classifiers could be included too. However, since SVM is already a classifier with a very high accuracy rate, we do not expect that there are classifiers available at the moment that would perform substantially better on their own. It is however possible that higher classification scores can be achieved with other normalisation methods than currently used in this work.

  There is a possibility that the auto-train functionality for the SVM parameters as explained in chapter 4 is not performing well enough and that there are other combinations of parameters available with which SVM could achieve higher scores, but we do not expect this to have a significant impact.

- **Bin size**: Even though the influence of histogram sizes on the classification accuracy has been tested in chapter 6, it has been made clear that a more detailed research on the number of histogram bins has to be completed before any conclusions on the optimal number of bins can be made. Even though the influence of the bin size seems minor in the overall accuracy score, every small improvement can help to attain an optimal classification result.

- **Performance improvements**: Seen the fact that the FGT must calculate all these features for every images and it still does this in only several seconds for each image, we can state that the performance of the application is sufficient. However, when the resolution of the images in the dataset increases, the performance of the tool drops significantly. Since the code for this research was mostly written for workability, research and testing, and not for ultimate efficiency, a very substantial performance boost could be gained by improving the codebase of the tool.

  One option to improve performance is to make the feature generation tool multi-threaded. An easy operation, since the set of images in a dataset can be seen as a batch of individual tasks, making the computational problem embarrassingly parallel. At the end the results of the different threads have to be combined into one .data file.

- **Featured and subset possibilities**: Since *Featured* is still a work in progress, some of the tests should be redone when the application is finished, which might result in finding better subsets of descriptors. This would be even more important if extra descriptors become available. It is also possible that a further developed *Featured* will give new opportunities for exploring the descriptor space.

9 CONCLUSION

All the research in this field is directed towards creating a viable classification-assistance-tool for medical professionals. We however have to conclude that this end goal is not yet reached. The long list of future work in the chapter 8 has shown that more research should be done first.

Whilst the automatic classification systems do reach accuracy results comparable to dermatologists, we have to emphasize that there are still too many aspects that have a negative effect on the results and therefore prevent the use of an automatic classification system in active medical service as an assistance tool.

Datasets, pre-processing, selection and quality of features, normalisation, type of classifier used: we have concluded in chapter 7 that all have their influences, large or small, on classification accuracy. Chapter 7 has also emphasized that there are many non-Computing Science related factors influencing classification outcome, especially in relation to the datasets.

Consequently we need to ask ourselves: have we actually answered our main research question?:

> How can we empower the designer of skin lesion classification tools to effectively and efficiently explore the design space of skin classification algorithms in order to design better such tools?

We have shown in this work that this is achievable by creating a platform in which descriptors of all kinds can be applied to datasets to generate measurements that available for storage in an intermediate format. With a visual analysis tool such as *Featured*, it is possible to explore the design space of descriptors and determine the influence of specific ones. Using this knowledge, it is possible to either classify all descriptor data or a subset thereof using any of several classifiers available.

Our highest accuracy score of 0.822 matches results achieved by others in the field. If any or all of the remarks explained in chapters 7 and 8 are pursued, the potential of creating an automatic skin lesion classification system that can assist in the current medical world will increase substantially.

ACRONYMS

ACM	Active Contour Model. 8
AI	Artificial Intelligence. 10
ANN	Artificial Neural Network. 11–13, 37
ANOVA	Analysis of Variance. 51
CIE	Commision Internationale de l'Eclairage. 19
CUDA	Compute Unified Device Architecture. 8
FGT	Feature Generation Tool. 16, 35, 37, 49, 52, 57, 59, 63, 71, 80
GPU	Graphics Processing Unit. 8
GUI	Graphical User Interface. 49
GVF	Gradient Vector Flow. 8, 9, 48, 65
HOG	Histogram of Oriented Gradients. iii, 25, 30, 31, 33, 46, 52, 57–60, 63, 79
IFT	Imaging Foresting Transform. 8, 9
kNN	k Nearest Neighbours. iii, 7, 11, 14, 40, 41, 46, 47, 53–58, 63, 65, 68, 69, 80
Lab	Lightness, a colour component and b colour component. 19–21, 26, 60
LBP	Local Binary Patterns. iii, 25, 70
LSP	Least Squares Projection. 50, 60, 61
LVQ	Learning Vector Quantisation. 7, 11, 41
PCA	Principal Component Analysis. 11, 34
RAW	Raw Image Format. 20
RBF	Radial Basis Function. 14, 45, 46, 80
RGB	Red, Green and Blue. 17, 19–22, 24, 26, 27, 60
SSD	Solid State Drive. 37
SVM	Support Vector Machines. iii, 8, 11, 14, 25, 30, 40, 41, 43–45, 71, 80
SVML	Support Vector Machine Linear. 46, 47, 53, 55–58, 63, 68, 69, 80
SVMR	Support Vector Machine Radial Basis Function. 46, 47, 53, 55–58, 63, 65, 68, 69, 80

BIBLIOGRAPHY

[1] P. Schmid-Saugeona, J. Guillodb, and J.-P. Thirana, "Towards a computer-aided diagnosis system for pigmented skin lesions," *Computerized Medical Imaging and Graphics*, vol. 27, no. 1, pp. 65–78, 2003.

[2] K. Korotkov and R. Garcia, "Computerized analysis of pigmented skin lesions: a review," *Artificial intelligence in medicine*, vol. 56, no. 2, pp. 69–90, 2012.

[3] A. H. Sam and J. T. Teo, *Rapid medicine*, vol. 18. John Wiley & Sons, 2011.

[4] Y. I. Cheng, R. Swamisai, S. E. Umbaugh, R. H. Moss, W. V. Stoecker, S. Teegala, and S. K. Srinivasan, "Skin lesion classification using relative color features," *Skin Research and Technology*, vol. 14, no. 1, pp. 53–64, 2008.

[5] A. A. Marghoob, L. D. Swindle, C. Z. Moricz, F. A. S. Negron, B. Slue, A. C. Halpern, and A. W. Kopf, "Instruments and new technologies for the in vivo diagnosis of melanoma," *Journal of the American Academy of Dermatology*, vol. 49, no. 5, pp. 777–797, 2003.

[6] P. Rauber, R. d. Silva, S. Feringa, M. Celebi, A. Falcao, and A. Telea, "Interactive image feature selection aided by dimensionality reduction," in *Proc. International EuroVis Workshop on Visual Analytics (EuroVA)*, 2015. accepted.

[7] H. L. Kaufman, *The melanoma book: a complete guide to prevention and treatment*. Gotham, 2005.

[8] A. F. Jerant, J. T. Johnson, C. Sheridan, and T. J. Caffrey, "Early detection and treatment of skin cancer," *American family physician*, vol. 62, no. 2, pp. 357–386, 2000.

[9] A. G. Manousaki, A. G. Manios, E. I. Tsompanaki, J. G. Panayiotides, D. D. Tsiftsis, A. K. Kostaki, and A. D. Tosca, "A simple digital image processing system to aid in melanoma diagnosis in an everyday melanocytic skin lesion unit. a preliminary report," *International journal of dermatology*, vol. 45, no. 4, pp. 402–410, 2006.

[10] J. Koehoorn, A. Sobiecki, D. Boda, A. Diaconeasa, S. Doshi, S. Paisey, A. Jalba, and A. Telea, "Automated digital hair removal by threshold decomposition and morphological analysis," in *International Symposium on Mathematical Morphology (ISMM)*, 2015. Accepted.

[11] A. Parolin, E. Herzer, and C. R. Jung, "Semi-automated diagnosis of melanoma through the analysis of dermatological images," *Proc. SIBGRAPI, 71–78*, 2010.

[12] J. H. Christensen, M. B. Soerensen, Z. Linghui, S. Chen, and M. O. Jensen, "Pre-diagnostic digital imaging prediction model to discriminate between malignant melanoma and benign pigmented skin lesion," *Skin Research and Technology*, vol. 16, no. 1, pp. 98–108, 2010.

[13] M. E. Celebi, H. A. Kingravi, B. Uddin, H. Iyatomi, Y. A. Aslandogan, W. V. Stoecker, and R. H. Moss, "A methodological approach to the classification of dermoscopy images," *Computerized Medical Imaging and Graphics*, vol. 31, no. 6, pp. 362–373, 2007.

[14] S. Feringa, S. L. Geertsema, M. Biehl, and M. Wilkinson, "Automatic separation of skin lesions from healthy human skin," tech. rep., University of Groningen, 2012.

[15] S. Feringa and M. Biehl, "Application of γDLVQ in an automatic skin lesion segmentation program," tech. rep., University of Groningen, 2012.

[16] Z. Yu, H.-S. Wong, and G. Wen, "A modified support vector machine and its application to image segmentation," *Image and Vision Computing*, vol. 29, no. 1, pp. 29–40, 2011.

[17] P. E. Rauber, A. X. F. ao, T. V. Spina, and P. J. de Rezende, "Interactive segmentation by image foresting transform on superpixel graphs," *SIBGRAPI*, 2013.

[18] A. C. Telea, *Data visualization: principles and practice*. CRC Press, second edition, 2014.

[19] J. Jans, R. Kiers, and A. C. Telea, "Automatic segmentation and visualization of skin lesions," tech. rep., University of Groningen, 2013.

[20] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[21] S. Choi and C. Kim, "Automatic initialization active contour model for the segmentation of the chest wall on chest ct," *Healthcare Informatics Research 16.1: 36–45. PMC.*, 2010.

[22] J. Tang, S. Millington, S. T. Acton, J. Crandall, and S. Hurwitz, "Ankle cartilage surface segmentation using directional gradient vector flow snakes.," in *ICIP*, pp. 2745–2748, IEEE, 2004.

[23] C. Xu and J. L. Prince, "Gradient vector flow: A new external force for snakes," *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 66–71, 1997.

[24] B. Kusumoputro and A. Ariyanto, "Neural network diagnosis of malignant skin cancers using principal component analysis as a preprocessor," *IEEE Transactions on Neural Networks, 0-7803-4859-1/98*, 1998.

[25] I. Maglogiannis and C. N. Doukas, "Overview of advanced computer vision systems for skin lesions characterization," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 13, no. 5, pp. 721–733, 2009.

[26] S. Dreiseitl, L. Ohno-Machado, H. Kittler, S. Vinterbo, H. Billhardt, and M. Binder, "A comparison of machine learning methods for the diagnosis of pigmented skin lesions," *Journal of biomedical informatics*, vol. 34, no. 1, pp. 28–36, 2001.

[27] T. Kohonen, "Self-organizing maps," *Springer-Verlag, Berlin*, 1995.

[28] E. Mwebaze, P. Schneider, F.-M. Schleif, J. R. Aduwo, J. A. Quinn, S. Haase, T. Villmann, and M. Biehl, "Divergence-based classification in learning vector quantization," *Neurocomputing*, vol. 74, no. 9, pp. 1429–1435, 2011.

[29] P. Schneider, *Advanced methods for prototype-based classification*. Atto Producties Europe, 2010.

[30] K. Bunte, *Adaptive dissimilarity measures, dimension reduction and visualization*. University Library Groningen, 2011.

[31] S. Russell and P. Norvig, *Artificial Intelligence, a modern approach*. Third edition, Pearson International, as Prentice-Hall, New Jersey, 2010.

[32] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[33] D. O. Hebb, *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[34] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, J. Suykens, and T. Van Gestel, *Least squares support vector machines*, vol. 4. World Scientific, 2002.

[35] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of clinical epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996.

[36] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification. 2nd," *Edition. New York*, 2001.

[37] M. Elbaum, A. W. Kopf, H. S. Rabinovitz, R. G. Langley, H. Kamino, M. C. Mihm, A. J. Sober, G. L. Peck, A. Bogdan, D. Gutkowicz-Krusin, *et al.*, "Automatic differentiation of melanoma from melanocytic nevi with multispectral digital dermoscopy: a feasibility study," *Journal of the American Academy of Dermatology*, vol. 44, no. 2, pp. 207–218, 2001.

[38] C. R. Jung and J. Scharcanski, "Sharpening dermatological color images in the wavelet domain," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 1, pp. 4–13, 2009.

[39] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[40] K. Pearson, *Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 186: 343–414, 1895.

[41] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[42] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. 1998.

[43] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification.," *IEEE Transactions on Systems, Man and Cybernetics 3 (6), 610-620*, 1973.

[44] R. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE 67, 786–804*, 1979.

[45] A. Materka and M. Strzelecki, "Texture analysis methods – a review," tech. rep., Institute of Electronics, Technical University of Lodz, 1998.

[46] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition* (C. Schmid, S. Soatto, and C. Tomasi, eds.), vol. 2, (INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334), pp. 886–893, June 2005.

[47] L. Shen, R. Rangayyan, and J. Desautels, "Application of shape analysis to mammographic calcifications," *Medical Imaging, IEEE Transactions on*, vol. 13, pp. 263–274, Jun 1994.

[48] A. Telea and J. J. Van Wijk, "An augmented fast marching method for computing skeletons and centerlines," in *Proceedings of the symposium on Data Visualisation 2002*, pp. 251–ff, Eurographics Association, 2002.

[49] L. Gupta and M. D. Srinath, "Contour sequence moments for the classification of closed planar shapes," *Pattern Recognition*, vol. 20, no. 3, pp. 267–272, 1987.

[50] P. Arques, R. Molina, M. Pujol, and R. Rizo, "Distance histogram to centroid as a unique feature to recognize objects," *Institute for Systems and Technologies of Information, Control and Communication*, 2006.

[51] P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 701–715, 1989.

[52] E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 272–285, 2007.

[53] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," vol. 2, pp. 524–531, 2005.

[54] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," vol. 1, no. 1-22, pp. 1–2, 2004.

[55] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[56] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[57] H. A. Song and S.-Y. Lee, "Hierarchical representation using nmf," pp. 466–473, 2013.

[58] S. Raschka, "Naive bayes and text classification 1, introduction and theory," 2014.

[59] E. Fix and J. H. Junior, "Discriminatory analysis, nonparametric discrimination: Consistency properties," *Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine*, 1951.

[60] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory, IT-13(1), 21-27*, 1967.

[61] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[62] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[63] P. Winston, "Support vector machines," *MIT OpenCourseWare lecture 16, course 6.034 Artificial Intelligence*, 2010.

[64] A. Aizerman, E. M. Braverman, and L. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and remote control*, vol. 25, pp. 821–837, 1964.

[65] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel Methods in Computational Biology*, pp. 35–70, 2004.

[66] A. Ng, "Support vector machines, using a svm," *Coursera Open Course Online, Standford University, course on Machine Learning*, 2011.

[67] G. Argenziano, H. P. Soyer, *et al.*, "Interactive atlas of dermoscopy," 2002.

[68] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz, "Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, no. 3, pp. 564–575, 2008.

[69] R. M. Martins, D. B. Coimbra, R. Minghim, and A. Telea, "Visual analysis of dimensionality reduction quality for parameterized projections," *Computers & Graphics*, vol. 41, pp. 26–42, 2014.

A

APPENDIX

A.1 FEATURE GENERATION TOOL IMPLEMENTATION DETAILS

The application of our selection of features $\mathcal{F}$ on a dataset of images, and the generation of the data output file used in further steps, is combined in the `featuregen` tool. It is available as a *Mac OSX* and *Linux* command line tool, written in *C++*. Software requirements are: `g++` version 4.8.0 or higher, `make`, `cmake` and *OpenCV* version 2.4.11 or higher. To run the the program on the dataset found in `orig/` with the labels `labels/` run:

```
./featuregen orig/ labels/ 16 16 border_features=yes ms1_16_16.data
```

Here we apply the tool on a dataset, setting its parameters with the following command line parameters:

1. The path to the original images, `orig/` in our example above.
2. The path to the mask images, `labels/` in our example above.
3. How many bins all histograms of the features should have, except the HOG descriptor.
4. How many bins the histogram of specifically the HOG descriptor should use.
5. Whether or not features using the skin lesion border should be used. Use either `border_features=yes` or `border_features=no`.
6. And finally the name of the output file must be specified, combined with the extension name.

It then stores all the filenames and paths of all images and mask images (also known as labels). If the number mask of images is unequal to the number of original images, the tool goes into the earlier explained mask mode and will generate a mask of the same resolution for each image instead of using the supplied mask image. This will be the same for the whole batch of images.

In each iteration of the main processing loop, an image and a corresponding mask image is imported using .ppm and .pgm read functions. P2, P5 (.pgm) and P5, P6 (.ppm) variants of the *Netbpm* format are supported. The two images are stored in

memory and consecutively each feature is extracted and stored in a feature vector. At the end of the iteration the feature vector is added to a list. When all images are processed, an output file is generated in the .data text file format from this list of feature vectors, in accordance with the standard as used by the *Data Exploration Tool* by [69]. This file can then be imported together with the original images in the FGT.

A.2 CLASSIFICATION TOOL IMPLEMENTATION DETAILS

To be able to apply the classifiers and normalisation methods on our feature data we have made a *Mac OSX* and *Linux* command line tool, written in *C++* called `classifier`. Software requirements are: textttg++ version 4.8.0 or higher, `make`, `cmake` and *OpenCV* version 2.4.11 or higher. Two example cases for how to run the program:

```
./classifier knn data/features.data 4 7 2
```

```
./classifier svm data/features.data 4 1 2
```

For which the tool has the following arguments:

- Type of classifier used. `knn` or `svm`.
- Path and name of the file with all the feature data.
- Number of feature elements on each feature vector that have to skipped because they are metadata (and should not be included in the classification).
- k value in case kNN is used, or the mode of operation in case SVM is selected. Mode 1 means Linear SVM, mode 3 results in the use of the RBF kernel.
- Normalisation mode. Mode 0 means no normalisation, mode 1 means scaling and mode 2 is for standard score normalisation.

The kNN implementation is made by the author himself, making it possible to fully control the functionality of the classifier. Since implementing SVM is complex, we opted to use the version present in *OpenCV*. SVM and its kernels are dependent on several parameters that have to be set before the classification process starts. Finding the optimal parameter settings is however an complicated affair.

The *OpenCV* implementation includes an auto-train feature for finding the best parameter settings. In case of the SVML, we can state that the operation is fast. However, SVMR needs a few seconds to find the optimal parameters. The tool at this moment prints the classification result as command line output. However, this can be easily extended to disclosing the results using output files.