Image Analysis of Malign Melanoma:

Wavelets and SVD

Master's Thesis in Applied Mathematics

Dan Dolonius

Department of Mathematical Sciences
UNIVERISTY OF GOTHENBURG
Gothenburg 2013

Image Analysis of Malign Melanoma:

Wavelets and SVD

Department of Mathematical Sciences
Dan Dolonius

Tutors:	Ivar	Gustafsson
	Mohammad	Asadzadeh
Examiner:	Larissa	Beilina

Department of Mathematical Sciences
UNIVERISTY OF GOTHENBURG
Gothenburg 2013

Acknowledgements

I would like to thank Mohammad Asadzadeh and Ivar Gustafsson for all their help, patience and overall pleasant discussions. Mats Rudemo for his lectures and help in image analysis. And finally John Paoli and all the people working with Dermatology and Venereology at Sahlgrenska for their help and data.

Abstract

In this thesis we investigate the use of Singular Value Decomposition and Wavelets for classification of malign melanoma. We first cover some basic theory, and later some tools and methods for image analysis such as morphological operators and thresholding. Finally, we make a simple linear model to evaluate the data. Although our model is not sufficient for classification, we manage to show trends in the data implying the possibility to refine the model to get better results.

Contents

1 Introduction

Melanoma is a type of skin cancer that, even though it is less common, probably is the most known and famous due to its high deadliness and correlation to sun bathing (UV-rays being the primary cause). With 1600000 new cases and 48000 deaths every year, melanoma is responsible for 75% of all the skin cancer related deaths[1]. It is of utter importance to be able to identify and remove the melanoma as fast as possible since the probability of relapse or spread is proportional to the depth of the lesion. If detected early the chance of a complete removal, and thus curing, is high.

Early inspection of lesions are often done visually using something called *Dermatoscopy*[2]. One then proceeds to look at features indicating whether the lesion is malign or not. There are a lot of features to consider and there is no single feature that one can rely on [14]. To compensate for this there is a framework called the ABCDE-rule (**A**ssymetry, **B**order, **C**olor, **D**iameter, **E**volution) which is to help classification using a score based system. Other frameworks are also used, but the basic principle behind all of them are pretty much the same. Using these frameworks, identification is still not a trivial task and it takes an experienced doctor in dermatology to be able to classify correctly over 90% of all cases.

To this date there are some classification tools and research regarding melanoma but unfortunately only a few or none of these have good enough results to be used in real life scenarios, due to the complex nature of the lesions. One big problem is that malign and benign lesions can share a lot of features, malign may look benign and vice versa. There are even cases when even a proffesional can not classify either case by just visually inspecting the lesion. This makes it tough to make computer aided classification. Partly because there are no direct indicators and also that some features are very small or discrete which make them hard to isolate.

With this in mind this thesis will take a shotgun approach to determine parameters; we fire a hail of bullets and see which hit. We thus may sacrify the quality of some parameters but make up for it in quantity (and we know that we will need quite a few parameters anyway).

This thesis will also try to focus on Wavelets and SVD approach for a couple of reasons. Wavelets have been used before for classifying malign cancer, where they are used to detect when the skewness and kurtosis become too high, since we know that healthy tissue will have a normal distribution, whereas malign does not [6].

[1]Wikipedia

[2]Basically looking at the lesion through a looking glass and a liquid medium a.k.a. dermoscopy, epiluminescence microscopy

2 Wavelet and MRA

Definition 1. *$\{\varphi_k\}_{k\in\mathbb{Z}}$ is a basis for a linear subspace V if any function $f \in V$ can be written uniquely as:*

$$f = \sum_k c_k \varphi_k.$$

□

With this, we define $\langle \cdot, \cdot \rangle$ as the scalar product and can then write any $f \in V$ as

$$f = \sum_k \langle f, \varphi_k \rangle \varphi_k. \tag{1}$$

Definition 2. *$L^2(\mathbb{R})$ is defined as the set of all functions $f(t)$ such that*

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty,$$

□

with the corresponding norm

$$\|f\| = \left(\int_{-\infty}^{\infty} |f(t)|^2 dt \right)^{1/2}.$$

Definition 3. *A basis $\{\varphi_k\}$ of a subspace V is said to be a Riesz basis if, for $f = \sum_k c_k \varphi_k$,*

$$A\|f\|^2 \leq \sum_k |c_k|^2 \leq B\|f\|^2,$$

holds, where $0 < A \leq 1 \leq B$ are some constants which do not depend on f.

□

Definition 4. *A multiresolution analysis (MRA) is a family of closed subspaces V_j of $L^2(\mathbb{R})$ with the following properties:*

- *$V_j \subset V_{j+1}$ for all $j \in \mathbb{Z}$;*
- *$f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1}$ for all $j \in \mathbb{Z}$;*
- *$\bigcup\limits_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$;*
- *$\bigcap\limits_{j\in\mathbb{Z}} V_j = \{0\}$;*
- *There exists a scaling function $\varphi \in V_0$ such that $\{\varphi(t-k)\}$ is a Riesz basis for V_0, where $t \in \mathbb{R}$ and $k \in \mathbb{Z}$.*

□

To make proper use of wavelets and MRA we want to impose some extra restrictions to the scaling functions, namely:

Property 1: Localized in time, which means a rapid decay to zero as $t \to \pm\infty$, or more preferably;

Property 2: Compact support: $t \notin (a, b)$ and $|a - b| < \infty \Rightarrow \varphi(t) = 0$;

Property 3: $\int_{-\infty}^{\infty} \varphi(t) dt = 1$.

The localization properties (Property 1) and (Property 2) will ensure us that our approximations will contain local information of f. The last, (Property 3), is closely related to approximations of f and vanishing moments (defined in section 2.2.4).

To make use of locality we add some more properties to the scaling function, namely *dilation* and *translation* ($j, k \in \mathbb{Z}$):

$$\varphi_{j,k}(t) = 2^{j/2} \varphi(2^j t - k).$$

Having the scaling function defined in this way will ensure that

$$\|\varphi_{j,k}\| = \|\varphi\|,$$

since

$$\|\varphi_{j,k}\| = \left(\int_{-\infty}^{\infty} |2^{j/2}\varphi(2^j t - k)|^2 dt\right)^{1/2} = \left(2^{-j}\int_{-\infty}^{\infty} 2^j |\varphi(t-k)|^2 dt\right)^{1/2} = \|\varphi\|.$$

From this it follows that $\varphi_{j,k}$ is a Riesz basis for V_j. Using (1) we can construct a function, $f_j \in V_j$, to approximate some function, $f \in V_j$, by:

$$f_j(t) = \sum_k s_{j,k}\varphi_{j,k}(t), \tag{2}$$

where $s_{j,k} = \langle f, \varphi_{j,k}\rangle$.

2.1 Wavelets

In the previous section we prepared us to approximate a function in different time scales. Now, to make our MRA more interesting we introduce *wavelets*. In the same way that scaling functions correspond to approximation, wavelet functions correspond to differences (or errors), that is; given the approximations $f_j \in V_j$ and $f_{j+1} \in V_{j+1}$ for $j \in \mathbb{Z}$, we denote the difference d_j between the two levels as

$$d_j = f_{j+1} - f_j,$$

or equivalently

$$f_{j+1} = d_j + f_j.$$

From this we get a recursive relation between different levels of approximations as

$$f_{j+1} = d_j + d_{j-1} + \ldots + d_{j_0} + f_{j_0}. \tag{3}$$

To be able to generalize we want our details to be expressed in some basis as we have done in (2), i.e.

$$d_j(t) = \sum_k w_{j,k}\psi_{j,k}(t),$$

where $w_{j,k} = \langle f, \psi_{j,k}\rangle$. This brings us to our next definition.

Definition 5. *For a general MRA, a function ψ is said to be a wavelet if the detail space $W_0 \subset L^2(\mathbb{R})$ spanned by the functions $\psi(t-k)$ complements V_0 in V_1. By this we mean that any $f_1 \in V_1$ can be uniquely written as $f_1 = f_0 + d_0$, where $f_0 \in V_0$ and $d_0 \in W_0$. We write this formally as $V_1 = V_0 \oplus W_0$. Finally, we require the wavelets $\psi(t-k)$ to be a Riesz basis for W_0.*

□

Here ψ is called the *mother wavelet*, and just as we imposed some restrictions to the scaling function, we impose some for our wavelet as well:

Property 1: Localized in time;

Property 2: Compact support;

Property 3: $\int_{-\infty}^{\infty} \psi(t)dt = 0$.

Note that these are the same properties as we want for the scaling function except that we want the integral to be equal zero. Similarly we define the dilated and translated wavelet function:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k),$$

and can thus safely write the approximation in (3), at an arbitrary level $J \in \mathbb{Z}$, as

$$f_J(t) = \sum_{j=j_0}^{J-1}\sum_k w_{j,k}\psi_{j,k}(t) + \sum_k s_{j_0,k}\varphi_{j_0,k}(t),$$

or formally expressed in its respective spaces [3] as

$$V_J = W_{J-1} \oplus W_{J-2} \oplus \ldots \oplus W_{j_0} \oplus V_{j_0}.$$

Further, looking at (3), we can show that $f_{j_0} \to 0$ when $j_0 \to -\infty$, as well as $\|f - f_j\| \to 0$ when $j \to \infty$ cf.[9]. Thus letting j tend to infinity we get

$$f(t) = \sum_{j,k} w_{j,k}\psi_{j,k}(t),$$

which we call the *wavelet decomposition* of f.

2.2 Wavelet decomposition

2.2.1 Properties of wavelets and scaling functions

We start with looking at definition 4. Condition 1. states that $V_0 \subset V_1$ which tells us that the scaling equation can be expressed as

$$\varphi(t) = 2\sum_k h_k \varphi(2t - k). \tag{4}$$

If we take the Fourier transform of this equation we get

$$\hat{\varphi}(\omega) = H(\omega/2)\hat{\varphi}(\omega/2),$$

where

$$H(\omega) = \sum_k h_k e^{-ik\omega}.$$

To understand how the scaling function reacts to different frequencies, we first let $\omega = 0$ to investigate what happens at the low frequencies. According to (Property 3), in the definition of scaling functions[4], $\hat{\varphi}(0) = 1$ and thus:

$$H(0) = \sum_k h_k = 1.$$

Correspondingly, for high frequencies, i.e. $\omega = \pi$ we get:

$$H(\pi) = 0.$$

So we keep low frequencies and discard high ones, thus we conclude that H is a low-pass filter. If we do the same procedure for the wavelet function we realize, using definition 5, that $W_0 \subset V_1$ and thus $\psi \in V_1$. We again use Condition 1 in definition 4 to get

$$\psi(t) = 2\sum_k g_k \varphi(2t - k),$$

with corresponding Fourier transform

$$\hat{\psi}(\omega) = G(\omega/2)\hat{\varphi}(\omega/2),$$

where

$$G(\omega) = \sum_k g_k e^{-ik\omega}.$$

Similarly, to investigate how the wavelet function reacts to different frequencies, we realize that according to (Property 3) in the definition of wavelets [5] that $\hat{\psi}(0) = 0$, resulting in:

$$G(0) = \sum_k g_k = 0$$

and

$$G(\pi) = 1.$$

This tells us that G is a high-pass filter.

[3] $V_j = W_{j-1} \oplus V_{j-1}$ need not be unique, since ψ and W_j depend on how we choose our high pass filter. However, having an orthogonal basis for our scaling function will enforce uniqueness.

[4] The extra restrictions on page 2

[5] The extra restrictions on page 3

2.2.2 Orthogonality and Biorthogonality

Orthogonality and biorthogonality are properties which can be added to the wavelet and scaling functions. As in most cases this will significantly improve and simplify many algorithms[6]. The definitions are quite straightforward, and we will not go in depth in this paper, but rather stating the definitions.

Definition 6. *The wavelet and scaling functions satisfy the orthogonality condition if*

- $\langle \varphi_{j,k}, \varphi_{j,l} \rangle = \delta_{k,l}$;
- $\langle \psi_{j,k}, \psi_{j,l} \rangle = \delta_{k,l}$;
- $\langle \varphi_{j,k}, \psi_{j,l} \rangle = 0$.

□

Where

$$\delta_{k,l} := \begin{cases} 1 & \text{if } k = l, \\ 0 & \text{else.} \end{cases}$$

In terms of coefficients this can be viewed as

- $\sum_l h_l h_{l+2k} = \delta_k/2$;
- $\sum_l g_l g_{l+2k} = \delta_k/2$;
- $\sum_m h_{m+2k} g_{m+2l} = 0$.

Biorthogonal systems are quite similar to orthogonal systems. The basic principle is that we have two sets of wavelet and scaling functions, the originals and the duals. We define them in a similar manner as we have done before; here $\tilde{\varphi}_{j,k}$ and $\tilde{\psi}_{j,k}$ denote the duals of scaling and wavelet functions, respectively:

$$\tilde{\varphi}(t) = 2 \sum \tilde{h}_k \tilde{\varphi}(2t - k)$$

and

$$\tilde{\psi}(t) = 2 \sum \tilde{g}_k \tilde{\psi}(2t - k).$$

Definition 7. *The wavelet and scaling functions satisfy the biorthogonality condition if*

- $\langle \varphi_{j,k}, \tilde{\varphi}_{j,l} \rangle = \delta_{k,l}$;
- $\langle \psi_{j,k}, \tilde{\psi}_{j,l} \rangle = \delta_{k,l}$;
- $\langle \varphi_{j,k}, \tilde{\psi}_{j,l} \rangle = 0$;
- $\langle \tilde{\varphi}_{j,k}, \psi_{j,l} \rangle = 0$.

□

Note that φ and ψ are not orthogonal to each other, but to their duals. Similarly, the duals are not orthogonal to each other, but to the originals.

2.2.3 The Discrete Wavelet transform

Since the MRA is recursive in nature it effects the wavelet transforms to generally be recursive as well. As we will see, when we derive our algorithm we need an entry point in form of the scaling coefficients. We also assume orthogonal wavelets. Thus let us start at level $j+1$ with corresponding coefficients $s_{j+1,k} = \langle f, \varphi_{j+1,k} \rangle$ and continue from there[7]. We now have our approximation:

$$f_{j+1}(t) = \sum_k s_{j+1,k} \varphi_{j+1,k}(t).$$

[6]as you will see in the next section

[7]In practice, we assume to have sample values from $g(2^{-(j+1)}k)$ from which we numerically compute our coefficients. This process in known as **pre-filtering**. Probably the easiest and most common way to do this is to simply choose the sampled values of the function as the coefficients, see [9].

Using what we got from definition 5, we can split f_{j+1} as

$$f_{j+1} = f_j + d_j$$

and equivalently

$$\sum_k s_{j+1,k}\varphi_{j+1,k}(t) = \sum_k s_{j,k}\varphi_{j,k}(t) + \sum_k w_{j,k}\psi_{j,k}(t).$$

Performing a scalar multiplication on both sides we get

$$\sum_k s_{j+1,k}\langle\varphi_{j+1,k},\varphi_{j,l}\rangle = \sum_k s_{j,k}\langle\varphi_{j,k},\varphi_{j,l}\rangle + \sum_k w_{j,k}\langle\psi_{j,k},\varphi_{j,l}\rangle.$$

Since $\varphi_{j,k}$ constitute an orthogonal basis we know that $\langle\varphi_{j,k},\varphi_{j,l}\rangle = \delta_{k,l}$ and $\langle\psi_{j,k},\varphi_{j,l}\rangle = 0$ and thus we have

$$s_{j,k} = \sum_l s_{j+1,l}\langle\varphi_{j+1,l},\varphi_{j,k}\rangle.$$

Now, using (4) we get

$$\begin{aligned}\varphi_{j,k}(t) =& 2^{j/2}\varphi(2^jt-k)\\ =& 2^{j/2}\Big(2\sum_m h_m\varphi((2^jt-k)2-m)\Big)(2^{-1/2})(2^{1/2})\\ =& 2^{1/2}\sum_m h_m 2^{(j+1)/2}\varphi(2^{j+1}-2k-m)\\ =& 2^{1/2}\sum_m h_m\varphi_{j+1,2k+m}(t).\end{aligned}$$

And since $\langle\varphi_{j+1,l},\varphi_{j+1,2k+m}\rangle = \delta_{l,2k+m}$, we can deduce that

$$\langle\varphi_{j+1,l},\varphi_{j,k}\rangle = 2^{1/2}\sum_m h_m\langle\varphi_{j+1,l},\varphi_{j+1,2k+m}\rangle = 2^{1/2}h_{l-2k}.$$

The same procedure for the wavelet coefficients yields:

$$s_{j,k} = 2^{1/2}\sum_l h_{l-2k}s_{j+1,l}$$

$$w_{j,k} = 2^{1/2}\sum_l g_{l-2k}s_{j+1,l}.$$

Thus we now have a recursive formula for calculating the coefficients. To illustrate this we draw a decomposition[8] diagram in figure 1.

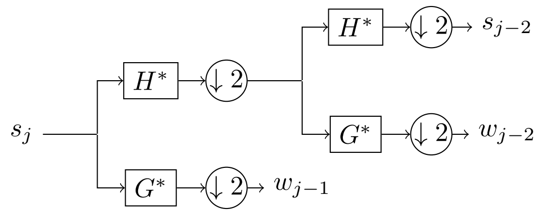


Figure 1: Wavelet decomposition.

If we want to reconstruct a signal from our coefficients, we need an *inverse wavelet transform*. We will not go in to more details in this paper but rather just present the *Fast Inverse Wavelet Transform* [9]. One of the advantages of using this transform is that we do not use the wavelet and scaling function explicitly, but rather the orthogonal filter bank. Given the scaling ($s_{j,k}$) and wavelet ($w_{j,k}$) coefficients [9] at level j, we have our reconstruction as

$$s_{j+1,k} = 2^{1/2}\sum_l (h_{k-2l}s_{j,k} + g_{k-2l}w_{j,k}),$$

[8]$\downarrow 2$ and $\uparrow 2$, stands for down sampling and up sampling by a factor of 2.

[9]recall that if we have done our decomposition correctly, we will have the wavelet coefficients for all levels up to the finest one: J

which we illustrate in figure 2. We realize that by using this we will finally end up with $s_{J,k}$, which are the coefficients at the finest level. Here we are basically done, but to recover the sample values we have to perform a post filtering step, which can be done by a convolution with the scaling function cf. [9].

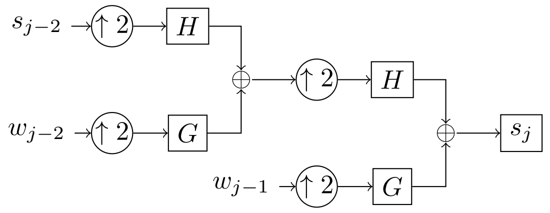


Figure 2: Wavelet reconstruction.

2.2.4 Vanishing moments

Definition 8. *A scaling function, φ, is said to have the approximation property if, for $\alpha \leq N-1$:*

$$\|f - f_j\| \leq C 2^{-j\alpha} \|f^{(\alpha)}\|$$

when it reproduces polynomials up to order $N-1$, i.e.:

$$\sum_k k^\alpha \varphi(t-k) = t^\alpha.$$

□

Abusing notation we write this as $t^\alpha \in V_j$, and thus $t^\alpha \perp \tilde{W}_j$, i.e. $\langle t^\alpha, \tilde{\psi}_{j,k} \rangle = 0$. With this in mind, we now define what we mean by vanishing moments for wavelet.

Definition 9. *A wavelet is said to have N vanishing moments if*

$$\int t^\alpha \psi_{j,k} dt = 0, \text{ for } \alpha = 0 \ldots N-1.$$

□

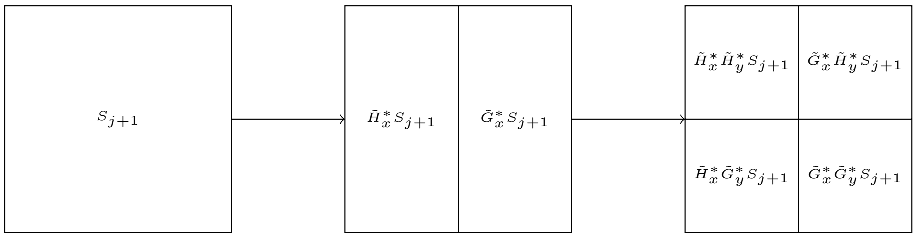
Vanishing moments is an important feature in wavelet theory. In combination with Fourier-transform it helps us to construct filter banks [9]. It is also what gives wavelets the ability to compress data. To illustrate this we consider a fine scale wavelet, $\psi_{j,k}$. Using (Property 1) or (Property 2) in the definition of wavelets, we can assume the wavelet to be zero outside an interval proportional to 2^{-j}. Given that the wavelet have N vanishing moments and that $\alpha \leq N$, if a signal would have continuous derivatives on this interval, we could approximate it by an $(\alpha-1)$-degree Taylor polynomial, $P_{\alpha-1}(t)$, with an error of order $\mathcal{O}(2^{-j\alpha})$, and thus:

$$\begin{aligned}\langle f, \psi_{j,k} \rangle &= \int f(t)\psi_{j,k}(t)dt \\ &= \int P_{\alpha-1}(t)\psi_{j,k}(t)dt + \mathcal{O}(2^{-j\alpha}) \\ &= \mathcal{O}(2^{-j\alpha}).\end{aligned}$$

We now know that the smooth parts of the signal will give small wavelet coefficients at finer scales, allowing us to store only the coefficients representing the parts containing a lot of detail, e.g. where there is abrupt changes. According to [7], the number of vanishing moments is weakly linked to how many oscillations the wavelets has, and in terms of signal and image processing, more than two vanishing moments is seldom desirable. The reason for this is that you do not really gain that much in quality, as much as that you would loose in speed for more vanishing moments. It is however more interesting and desirable when dealing with numerical analysis [7].

2.3 Two Dimensional Wavelet Transform

The two dimensional wavelet transform is a straightforward generalization of the one dimensional case. If we in the one dimensional case had our first sampled data as a vector, we will now start with a matrix, M as our samples instead, where $M_{i,j} = f(x_i, y_j)$. To do our decomposition we simply first transform all the rows to an approximation part and a detail part, and then do the same procedure for the resulting matrix but transforming the columns instead. [10]



s_{j+1}
$\tilde{H}_x^* s_{j+1}$
$\tilde{G}_x^* s_{j+1}$
$\tilde{H}_x^* \tilde{H}_y^* s_{j+1}$
$\tilde{G}_x^* \tilde{H}_y^* s_{j+1}$
$\tilde{H}_x^* \tilde{G}_y^* s_{j+1}$
$\tilde{G}_x^* \tilde{G}_y^* s_{j+1}$

Figure 3: 2D Wavelet decomposition.

With this in mind it makes sense to define our 2D mother functions as follows:

$$\Phi(x, y) := \varphi(x) \circ \varphi(y),$$

$$\Psi^H(x, y) := \varphi(x) \circ \psi(y),$$
$$\Psi^V(x, y) := \psi(x) \circ \varphi(y),$$
$$\Psi^D(x, y) := \psi(x) \circ \psi(y).$$

Here Φ is the scaling function and Ψ^H, Ψ^V, Ψ^D are our wavelet functions, which we will call the horizontal, vertical and diagonal details. Further we introduce our translated and dilated functions as:

$$\Phi_{j,k}(x, y) := 2^j \Phi(2^j x - k_x, 2^j y - k_y),$$
$$\Psi^H_{j,k}(x, y) := 2^j \Psi^H(2^j x - k_x, 2^j y - k_y),$$
$$\Psi^V_{j,k}(x, y) := 2^j \Psi^V(2^j x - k_x, 2^j y - k_y),$$
$$\Psi^D_{j,k}(x, y) := 2^j \Psi^D(2^j x - k_x, 2^j y - k_y).$$

Using basically the same definition as in the 1D case, we define our approximations spaces V_j as the set of all functions that can be written as:

$$f_j(x, y) = \sum_k c_{j,k} \Phi_{j,k}(x, y).$$

We end up with our scaling function as:

$$\Phi(x, y) = 4 \sum_k h_k \Phi(2x - k_x, 2y - k_y).$$

And our mother wavelets for $\alpha = \{H, V, D\}$

$$\Psi^\alpha(x, y) = 4 \sum_k g_k^\alpha \Phi(2x - k_x, 2y - k_y).$$

From this we get that $V_j \subset V_{j+1}$ and $W_j^\alpha \subset V_{j+1}$. We now know that any $f_j \in V_j$ can be expressed as

$$f_j = f_{j-1} + d^H_{j-1} + d^V_{j-1} + d^D_{j-1},$$

and once again, we have a recursive scheme for the wavelet decomposition. We can represent this to be a bit more intuitive in figure 4, where we denote S as the samples (or scaling coefficients) and W as the wavelet coefficients. Looking at this we also realize the potential for compression. If we assume that the squares represent a block of memory we realize that for storing just the coefficients, we do not really need to allocate any additional memory.

[10] The method used in this section, defines what is called separable wavelets. We won't go in more detail in this paper, but just note that not all wavelets in higher dimensions need to be defined in this way. We also assume a biorthogonal system.

Also, assuming that we can disregard the finest details (which are often polluted with too much noise anyway), we can directly disregard 3/4 of the required memory.

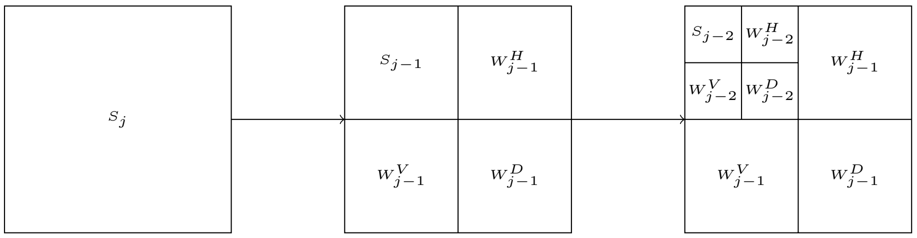


Figure 4: Two steps of decomposition matrix.

2.3.1 Border conditions of finite length signals

In the theory so far all signals has been assumed to have infinite length, and thus a problem arises when we have signals of finite length. We recall that to decompose a signal we need to compute $s_{j,k} = 2^{1/2} \sum_l h_{l-2k} s_{j+1,l}$, which can be problematic at the endpoints of the signal. To illustrate: Say, that we have a discrete signal $s = \{s_1,\ s_2\ \ldots\ s_{n-1},\ s_n\}$. To sample a certain element s_i where $0 << i << n$, we need to know s_{i-1}, s_i and s_{i+1}, both of which are present in our signal. Now, if we needed to sample s_1, we would need s_0, which does not exist. The solution to this is to simply extend the signal in some way, so that we could access s_0. In [5], we are introduced to some methods to extend the signal, namely:

1. **Extend by zeros** - Simply pad the extension by zeros. Simple, but does generally introduce jumps in the function.

2. **Symmetric extension** - Reflect the signal at the endpoints. Does not suffer from jumps in the function, but rather jumps in the first derivative.

3. **Periodic extension** - Repeat the signal at the endpoints. Good choice if the functions are periodic, or close to periodic.

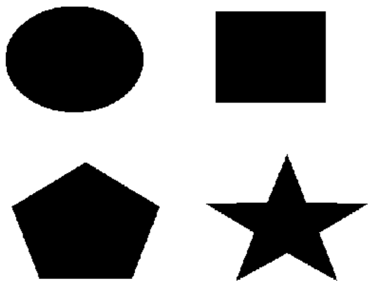
4. **Extrapolation** - Extrapolate the signal. e.g. constant, linear or quadratic extrapolation.

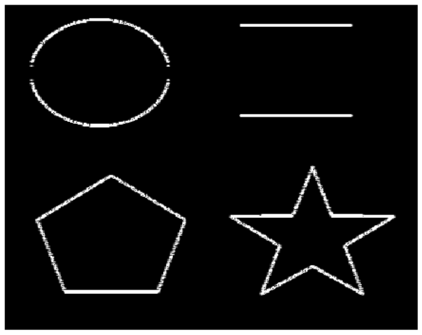
All methods have different strength and weaknesses. Zero padding is easy, needs no extra memory, but suffers from jumps in the function. Extrapolation by higher order polynomials reduces jumps in the function and its derivatives, proportional to the degree of the polynomial. It can yield a better extension in terms of accuracy, but with the disadvantage that we need more in terms of computations and/or memory. Symmetric and periodic extension is a good compromise, since we can at least reduce the jumps to the first derivative, as well as no extra memory requirements are needed since all necessary data is already included in the signal.

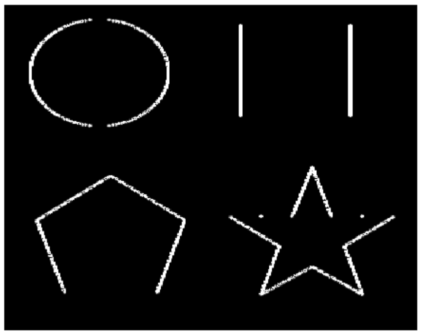
2.3.2 Edge detection and some other practical examples

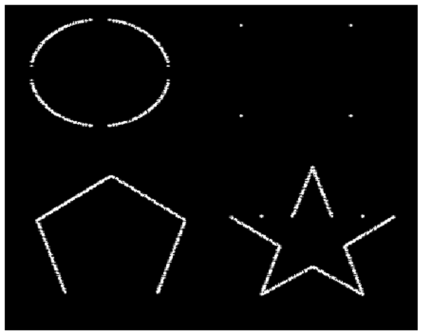
Besides just compression, the two dimensional wavelets can be used for many things in image processing such as edge detection. The basic principle is illustrated [11] in figure 5, where we first do one level of decomposition to get $S_{j-1}, W^H_{j-1}, W^V_{j-1}$ and W^D_{j-1}. We then disregard the scaling coefficients and reconstruct the image using just the details.

[11] In this example we reconstruct each detail separately, sum them together, and then clamp the result.


(a) Original image


(b) Horizontal details


(c) Vertical details


(d) Diagonal details

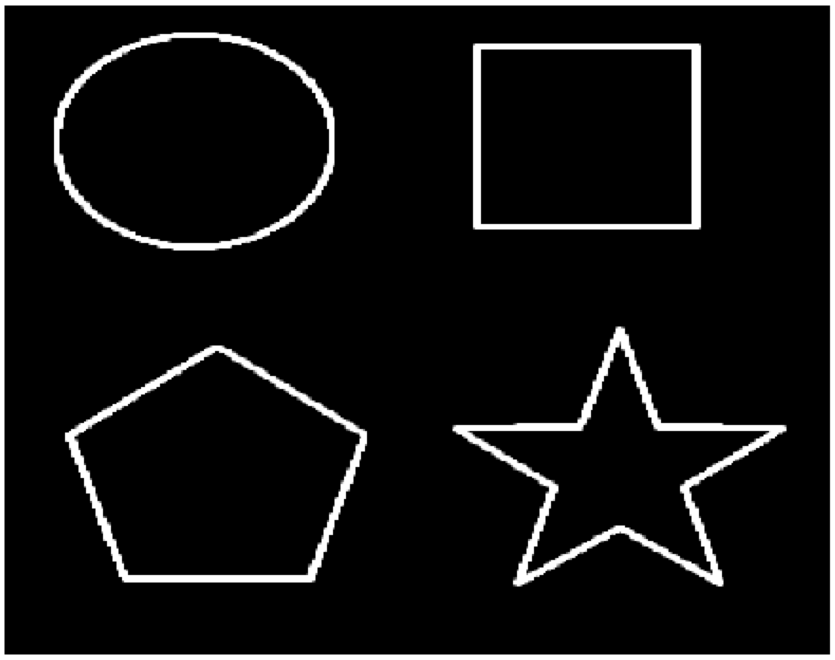

(e) Reconstructed details

Figure 5: Edge detection using daubechies2 filter bank.

There are many other beneficial uses as well, such as:

- Compression - Described in previos section
- De-noising - Similar to compression, threshold desired frequencies
- Sampling - In [11] they use wavelets to accelerate a ray traced scene. They do this by building importance maps [12] under different resolutions, from which they then sample from. Both steps are done using wavelets.
- Computer graphics in general - Can also be used for *accelerating by approximating* dynamic shadows in a rasterized 3D-scene, or as a complement to spherical harmonics in global illumination.
- Principal components
- Numerical analysis

It is easy to realize that wavelets in its own right is a quite powerful tool, but by combining it with other methods it can be even more useful.

[12] Basically a variance map

3 Singular Value Decomposition and Principal Component Analysis

3.1 Eigen Decomposition

From the theory regarding eigenvalues and eigendecomposition, we know that given a symmetric matrix $A \in \mathbb{R}^{m\times m}$ we have the decomposition

$$A = U\Lambda U^T,$$

where U is orthogonal i.e. $UU^T = U^TU = I$, and I is the identity matrix. Λ is a diagonal matrix containing the eigenvalues of A, and the i^{th} column of U is the eigenvector of A with eigenvalue $\Lambda_{i,i}$. To illustrate this, we multiply the decomposition by U from the right

$$AU = U\Lambda U^TU = U\Lambda.$$

Letting $\lambda_i = \Lambda_{i,i}$, and u_i be th i^{th} column of U we see that

$$Au_i = \lambda_i, \tag{5}$$

which we recognize as the common *eigenvalue equation*.

3.2 Singular Value Decomposition - SVD

Now, suppose $A \in \mathbb{R}^{m\times n}$ where $m \geq n$ and with rank $r \leq n$, we then end up with the decomposition [13]

$$A = U\Sigma V^T,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are both orthogonal matrices. Further $\Sigma \in \mathbb{R}^{m\times n}$ is a quasidiagonal matrix where $\Sigma_{i,i} \geq 0$ called the singular values, which are closely related to eigenvalues. To show this we multiply A with A^T from the right, to end up with A^TA being symmetric, and since A^TA is of rank[14] r we know that we will have r eigenvalues that are nonzero. Until stated otherwise we will be looking at the eigenvectors that correspond to the nonzero eigenvalues. We can now decompose A^TA as:

$$A^TA = V\Lambda V^T.$$

Repeating the same procedure as in (5), dropping the indexation for convenience, we see that

$$A^TAv = \lambda v,$$

where v is the eigenvector of A^TA with corresponding eigenvalue λ. If we once more multiply by A we see that

$$AA^TAv = \lambda Av,$$

implying that if v is and eigenvector for A^TA, then Av is an eigenvector for AA^T with the same eigenvalue, λ. We also realize that since we require v to be normalized, $\|Av\|_2^2 = \lambda$, since

$$(Av)^T(Av) = v^TA^TAv = v^T\lambda v = \lambda. \tag{6}$$

With this in mind, we now define $u_i := Av_i/\sigma_i$, where σ_i is the square root of λ_i. We realize that by replacing Av with Av/σ in equation (6), that this is a set of vectors which are pairwise orthogonal as well as normalized , and thus we can now write:

$$u_i^TAv_j = (A\frac{v_i}{\sigma_i})^TAv_j = \frac{v_i^TA^TAv_j}{\sigma_i} = v_i^Tv_j\frac{\lambda_i}{\sigma_i} = \begin{cases} \sigma_i & \text{if } i = j, \\ 0 & \text{else.} \end{cases} \tag{7}$$

We thus realize that if we write in matrix notation, we get:

$$U^TAV = \Sigma,$$

where Σ is composed of the singular values $(\sigma_1 \dots \sigma_r)$ on the diagonal and zero elsewhere. Now we are almost done, however since $U^T \in \mathbb{R}^{r\times m}$ and $V \in \mathbb{R}^{n\times r}$ they are still not orthogonal. To fix this we simply add the remaining $m-r$ and $n-r$ unit vectors, $\tilde{u}$ and $\tilde{v}$, to U and V respectively such that $(u_1 \dots u_m)$ and $(v_1 \dots v_n)$ are pairwise orthogonal,

[13]for $n \geq m$, simply transpose the matrix and do the same procedure. Alternatively pad with zeros such that $n = m$.

[14]This is also the rank of A and AA^T

and letting the corresponding singular values be zero i.e. $\Sigma \in \mathbb{R}^{m \times n}$ where $\Sigma_{i,i} = \sigma_i$ if $i \leq r$ and 0 otherwise. We now know that $U^T U = UU^T = I_{m,m}$ and $V^T V = VV^T = I_{n,n}$ and by multiplying by U from the left and V^T from the right, we thus acquire:

$$A = U\Sigma V^T,$$

which is our desired decomposition also known as *full SVD*.

Since we have some redundancy in our singular vectors where $\sigma = 0$, we get no information out of them and might as well just ignore those values using just the r non-redundant columns of U and V respectively, as well as letting $\Sigma_r = \operatorname{diag}(\sigma_i \dots \sigma_r)$ we end up with

$$A = U_r \Sigma_r V_r^T,$$

where $U_r \in \mathbb{R}^{m \times r}$ and $V_r \in \mathbb{R}^{n \times r}$, which is commonly called the *economy-* or *compact SVD*. If we, say take the k largest singular values with corresponding singular vectors we call that the *thin SVD*, which play a big role in compression.

3.3 Algorithm for decomposition

We have previously shown the existence of the SVD decomposition, and this section will explain some basic algorithms for doing this. There are many ways of doing the decomposition with varying pros and cons, as well as complexity. There is however a general structure[15] to these algorithms for a general $(m \times n)$ matrix, A:

1. Reduce A to bidiagonal form, such that $A = U_1 B V_1^T$, where U_1 and V_1 are orthogonal. This can be done by *Householder reflections*, which will be discussed in the next section.
2. Find the SVD of B, such that $B = U_2 \Sigma V_2^T$.
3. Let $U = U_1 U_2$ and $V = V_1 V_2$, and we have our desired decomposition $A = U\Sigma V^T$.

The reason for reducing A to bidiagonal form is that it's easier to find the SVD of such a matrix.

3.3.1 Householder Reflections

A Householder reflection is a transformation used on matrices to zero out parts of a vector x by reflecting it in a hyperplane perpendicular to a vector u with the condition $\|u\|_2 = (u^T u)^{1/2} = 1$. Since the reflection in vector notation can be written as:

$$x - 2\langle u, x\rangle u = x - 2uu^T x,$$

we realize that the corresponding transformation matrix can be defined as $P := I - 2uu^T$, which also is orthogonal. To show orthogonality we write:

$$PP^T = (I - 2uu^T)(I - 2uu^T)^T = I - 4uu^T + 4(uu^T)^2 = I.$$

Suppose now that we want to transform a vector x, such that $Px = [c, 0 \dots 0]^T = ce_1$. By expanding Px we see that u is a linear combination of x and e_1

$$Px = x - 2u(u^T x) = ce_1 \Rightarrow u = \frac{1}{2(u^T x)}(x - ce_1).$$

Also since P is a reflection, we know that

$$\|x\|_2 = \|Px\|_2 = |c|, \text{ and } c = \pm\|x\|_2.$$

Thus u must be parallel to

$$\tilde{u} := x + ce_1,$$

as well as that $u = \frac{\tilde{u}}{\|\tilde{u}\|_2}$. To keep cancellation to a minimum, we want the first entry of $x - ce_1$ as far from zero as possible and thus we choose $c = \operatorname{sign}(x_1)\|x_1\|e_1$, which gives:

$$\tilde{u} = x + \operatorname{sign}(x_1)\|x\|e_1.$$

We now have our $\tilde{u}$ as:

$$\tilde{u} = \begin{bmatrix} x_1 + \operatorname{sign}(x_1)|c| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

[15] with some exceptions such as the *Jacobi method*

To illustrate why this is useful we will repeat the same example as in [4] where they use Householder reflections to transform a matrix A to an upper triangular matrix. However, to show generality, we show this on a non quadratic matrix A, where $\bullet$ denotes an arbitrary real number.

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}.$$

$$\text{Choose } P_1 \text{ such that} \qquad P_1A = A_1 = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix}.$$

$$\text{Choose } P_2' \text{ such that} \qquad P_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & P_2' \end{array}\right], \qquad \text{where } P_2A_1 = A_2 = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix}.$$

$$\text{Choose } P_3' \text{ such that} \qquad P_3 = \left[\begin{array}{cc|c} 1 & & \\ & 1 & 0 \\ \hline \multicolumn{2}{c|}{0} & P_2' \end{array}\right], \qquad \text{where } P_3A_2 = A_3 = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix}.$$

$$\text{Choose } P_4' \text{ such that} \qquad P_4 = \left[\begin{array}{ccc|c} 1 & & & \\ & 1 & & 0 \\ & & 1 & \\ \hline \multicolumn{3}{c|}{0} & P_4' \end{array}\right], \qquad \text{where } P_4A_3 = A_4 = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \end{bmatrix}.$$

$$\text{Choose } P_5' \text{ such that} \qquad P_5 = \left[\begin{array}{cccc|c} 1 & & & & \\ & 1 & & & \\ & & 1 & & 0 \\ & & & 1 & \\ \hline \multicolumn{4}{c|}{0} & P_5' \end{array}\right], \qquad \text{where } P_5A_4 = A_5 = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & 0 & \bullet \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

So if we let $P = P_4P_3P_2P_1$, and so our transformation $A_5 = PA$ is an upper triangular matrix.
Since the main goal of the householder transformation was to get a bidiagonal structure we realize that in the same procedure can be used to reflect the rows as well, we just find a matrix by which we multiply from the right.
Choose L_i and then R_i such that:

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix},$$

$$L_1A = \begin{bmatrix} \bullet & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix} \text{ and } \quad (L_1A)R_1 = A_1 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix},$$

$$L_2A_1 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix} \text{ and } \quad (L_2A_1)R_2 = A_2 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix},$$

$$L_3A_2 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix} \text{ and } \quad (L_3A_2)R_3 = A_3 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & 0 \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \\ 0 & 0 & 0 & {\color{red}\bullet} & {\color{red}\bullet} \end{bmatrix},$$

$$L_4A_3 = A_4 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & 0 \\ 0 & 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \\ 0 & 0 & 0 & 0 & {\color{red}\bullet} \end{bmatrix} \text{ and finally } \quad L_5A_4 = \begin{bmatrix} \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & 0 \\ 0 & 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & 0 & \bullet \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

So in this case we get $L = L_1L_2L_3L_4L_5$ and $R = R_1R_2R_3$, which gives us our desired transformation $B = A_5 = LAR$. Changing notation as $U^T = L$ and $V = R$ we recognize our desired bi-diagonalization [16]:

$$B = U^TAR \Rightarrow A = UBV^T.$$

[16]This is not completely true, since we need to add extra columns so that U and V are indeed orthogonal.(as we did for proving the existence of the SVD)

3.3.2 Putting it all together

In the previous section we constructively proved the existence of the SVD. We here present some more practical information according to [4].

Lemma 1. *Let $B \in \mathbb{R}^{n \times n}$ be a bidiagonal matrix with $(a_1 \dots a_n)$ on the main diagonal and $(b_1 \dots b_{n-1})$ on the superdiagonal. There are three ways to convert the problem of finding the SVD of B to finding the eigenvalues and eigenvectors of a symmetric tridiagonal matrix, namely:*

1. *Let $A = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$ and $P = [e_1, e_{n+1}, e_2, e_{n+2} \dots e_n, e_{2n}]$, where e_i is the i:th column of the $(2n \times 2n)$ identity matrix. $T_{ps} = P^T AP$ is symmetric and tridiagonal* [17]*. T_{ps} has all zeros on the main diagonal and the super- and subdiagonal is $(a_1, b_1, a_2, b_2 \dots b_{n-1}, a_n)$. If $T_{ps}x_i = \alpha_i x_i$ is an eigenpair for T_{ps} and x_i being a unit vector, then $\alpha_i = \pm\sigma_i$, where σ_i is a singular value of B, and $Px_i = \frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$, where u_i and v_i are the left and right singular vectors of B.*

2. *BB^T is symmetric tridiagonal with maindiagonal $a_1^2 + b_1^2, a_2^2 + b_2^2 \dots a_{n-1}^2 + b_{n-1}^2, a_n^2$, and super- and subdiagonal $a_2 b_1, a_3 b_2 \dots a_n b_{n-1}$. The singular values of B are the square roots of the eigenvalues of BB^T, and the left singular vectors of B are the eigenvectors of BB^T. Also BB^T contains no information about the right singular vectors of B.*

3. *$B^T B$ is symmetric tridiagonal with maindiagonal $a_1^2, a_2^2 + b_1^2, a_3^2 + b_2^2 \dots a_n^2 + b_{n-1}^2$, and super- and subdiagonal $a_1 b_1, a_2 b_2 \dots a_{n-1} b_{n-1}$. The singular values of B are the square roots of the eigenvalues of $B^T B$, and the right singular vectors of B are the eigenvectors of $B^T B$. Also $B^T B$ contains no information about the left singular vectors of B.*

From this we see that algorithms such as *QR-iteration*, *divide-and conquer* and *Bisection with inverse iteration*, using a brute force approach can provide us with the singular values and maybe only the left or right singular vectors [4]. There is however, as always, drawbacks. First of all using this approach we need to compute **all** [18] the singular values, and thus ignoring some crucial benefits of the SVD. Also, forming $B^T B$ and/or BB^T can be quite numerically unstable, and we can thus loose precision, especially for small singular values. Due to this fact, we'd rather use algorithms that doesn't depend on $B^T B$ or BB^T, but rather on B or T_{ps} directly. Again referring to [4], there are some well known ways of doing this, namely:

- *QR iteration and its variations* - High accuracy and Fastest for finding all singular values for matrices of dimension roughly $n \leq 25$
- *divide-and-conquer*- High accuracy and Fastest for finding all singular values for matrices of dimension $n \geq 25$
- *Bisection and inverse iteration* - High accuracy for finding singular values within a desired interval, may suffer loss of orthogonality
- *Jacobi's method* - Overall high accuracy

3.4 Principal Component Analysis - PCA

Principal component analysis is defined as a linear orthogonal transformation where the greatest variance by any projection end up along the first basis in the new coordinate system, the second greatest along the second basis and so forth. The brute force approach is to recursively find the current axis orthogonal to previous axes which maximize the variance, which can be quite cumbersome and computationally expensive for large data sets. To improve this, one realized that this could be done by taking the eigendecomposition of the covariance matrix, however as we will see this will involve computing XX^T, for some matrix X. Say that we have n features of equal length m, we write the i^{th} feature as $x_i = \{a_1^i \dots a_m^i\}$ which is stored as a row vector. We define our mean vector

$$\bar{x}_i := \frac{1}{m} \sum_{k=1}^{m} a_k^i.$$

[17]ps stands for perfect shuffle, that is just a reordering of the matrix
[18]more than often only a few are desired

We can then define our covariance matrix:

$$C := \begin{bmatrix} Cov(x_1, x_1) & Cov(x_1, x_2) & Cov(x_1, x_3) & \dots \\ Cov(x_2, x_1) & Cov(x_2, x_2) & Cov(x_2, x_3) & \dots \\ Cov(x_3, x_1) & Cov(x_3, x_2) & Cov(x_3, x_3) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where $Cov(x_i, x_j)$ is calculated by:

$$Cov(x_i, x_j) = \frac{1}{m-1}\sum_{k=1}^{m}(a_k^i - \bar{x}_i)(a_k^j - \bar{x}_j).$$

To make notation easier we define the matrix $\hat{X}$:

$$\hat{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}.$$

And finally the zero mean matrix, where we simply subtract the sample mean from the corresponding sample:

$$X_{i,j} := \hat{X}_{i,j} - \bar{x_j}.$$

We can then simply write the covariance matrix as:

$$C = \frac{1}{m-1}XX^T.$$

Now we want to find our transformation matrix, T, which will be the projection onto our principal components (the eigenvectors), i.e. $Y = TX$, where Y is the transformed data. As just stated Y is projected onto an orthogonal basis, and so it will have zero covariance, i.e. $\frac{1}{m-1}YY^T$ will be a diagonal matrix D_Y. To start off we expand D_Y

$$\begin{aligned} D_Y =& \frac{1}{m-1}YY^T \\ =& \frac{1}{m-1}(TX)(TX)^T \\ =& \frac{1}{m-1}T(XX^T)T^T \\ =& TCT^T. \end{aligned}$$

We thus search for a matrix T such that TCT^T is diagonal. To find this matrix, our choice of T is such that $C = \frac{1}{m-1}XX^T = T^TDT$, where T^TDT is the eigendecomposition[19]. And we get the following

$$\begin{aligned} TCT^T =& T(T^TDT)T^T \\ =& (TT^T)D(TT^T) \\ =& D, \end{aligned}$$

and we see that T does indeed diagonalize TCT^T. We also note that the pricipal components are the rows of T (the eigenvectors of XX^T), and the i^{th} eigenvalue is the variance along the i:th principal component. To improve this algorithm, we realize that we don't have to compute XX^T if we use SVD instead since we can write, using the proper U:

$$Y = U^TX = U^TU\Sigma V^T = \Sigma V^T,$$

and

$$YY^T = \Sigma V^T(\Sigma V^T)^T = \Sigma(V^TV)\Sigma^T = \Sigma\Sigma^T,$$

which we realize is what we want since the singular values are the square roots of XX^T, and since Σ is diagonal $\Sigma\Sigma^T = \text{diag}(\sigma^2 \dots)$.

[19]Remember that T is orthogonal

3.4.1 Practical uses

SVD and PCA have many uses in numerical analysis, but also in statistics and as a tool as well. When dealing with unknown data of higher dimensions, we could use PCA to find the different clusters when the high dimension makes the data hard to visualize. Since we know that the further away the clusters are from each other would increase the variance in the corresponding direction, we could project the data on the major principal components and thus make visualization of the different cluster easier. In the same manner PCA is also used in statistical regression, since if we have too many variables in terms of data, we could just re-project the data and then choose only the components with the highest singular values. Yet another similar, but also different example is what we use in this paper. To effectively disregard of useless information in the pictures of the lesion we use PCA to figure out what angle we need to rotate the image such that it would fit an axis aligned box most efficiently.

4 Image analysis

Due to the nature of the images, it is very hard to isolate an optimal set of features, therefore the approach in this paper aims at collecting a large set of features that may or may not be significant. After the features have been collected stepwise linear regression is used to extract the features which are of highest relevance. From this much can be done, such as logistic regression, neural networks or support vector machines. However, here we simply look at the linear model to see if classification is plausible. The first problem with the images provided was that, due to the hardware used, we have our area of interest in a circular region of our image. This makes segmentation through thresholding tough, since the histogram will most likely be skewed or contain multiple minimas. This can however be solved by a morphological operator, *clear to border* which is a special case of *morphological reconstruction*, which is explained in the next section. We also take time here to define the LAB color space, designed by CIE[20]. Similar to HSV[21], and CMYK[22], LAB represents an image with different bases in terms of color. Here L represent luminosity (dark to light), A is green to magenta, and B is blue to yellow. The reason for investigating both RGB[23] and LAB is that while RGB has a linear distance in computational measures, LAB is designed to have linear distance regarding how the three cone cells in the human eye interpret colors[24] . How to convert an RGB image to LAB is not very difficult, but quite tedious; first we need to transform the RGB image to an *absolute color space* such as sRGB[25]. After this step the transformation basically involves some polynomial transformations from sRGB to XYZ[26], and then from XYZ to LAB. Here XYZ is another color space defined by CIE, which was one of the first color spaces designed to be analogous to the three cone cells in the human eye. It is also the basis for most other color spaces.

4.1 Preprocessing

4.1.1 Morphological Operators

Morphological operators are tools to clean up and manipulate images. They are easiest to define and explain in binary images, however the extension to gray scale images are quite straight-forward. A morphological operator is defined as a set of pixels, called the *structure element*, relative to a reference pixel. It's usually a simple neighborhood such as a (3×3) square or a disk, with the reference pixel in the center. Given an image A with structure element S and S' being S rotated around its reference pixel by 180°, as well as the complements A^c and S^c, we define the basic operators as:

Erosion operator

$$A \ominus S := \{(i,j) : S_{i,j} \subseteq A\}$$

Dilation operator

$$A \oplus S := (A^c \ominus S)^c$$

Opening operator

$$A \circ S := (A \ominus S) \oplus S'$$

Closing operator

$$A \bullet S := (A \oplus S) \ominus S'.$$

For a more intuitive definition we illustrate these operators in Figure 6 [27]. For the gray scale case we view the image and the structure element as functions, $A(x,y)$ and $S(x,y)$, where x,y denotes a pixel. We then define erosion as:

$$(A \ominus S)(x,y) := \min_{(i,j)\in S} (A(x+i, y+j) - S(i,j))$$

and dilation:

$$(A \oplus S)(x,y) := \max_{(i,j)\in S} (A(x-i, y-j) + S(i,j)).$$

[20] *The International Commission on Illumination*

[21] H: Hue, S: Saturation, V: Value(Brightness).

[22] C: Cyan, M: Magenta, Y: Yellow, K: Key(Black). Referring to the inks used in a color printer.

[23] R: Red, G: Green, B: Blue.

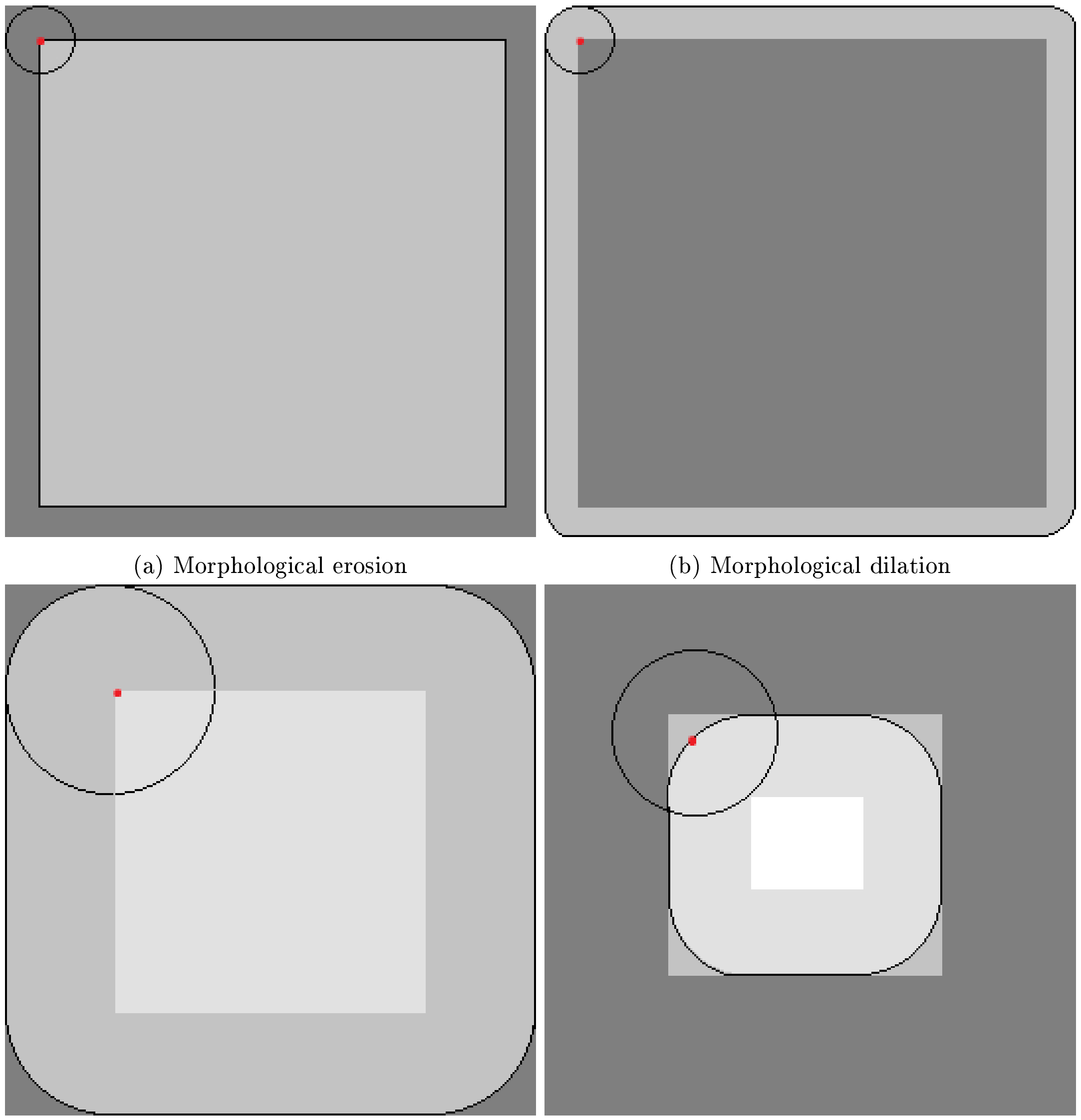
[24] Spectral sensitivity in short (S, 420-440 nm), middle (M, 530-540 nm), and long (L, 560-580 nm) wavelengths.

[25] As RGB, but with an extended Gamut. Gamut is the portion of the color space which can be represented or reproduced, e.g. pure red cannot be expressed in the CMYK color space.

[26] X: A mix of S and L , Y: Luminosity (Basically a wider M skewed towards L), Z: Equivalent to S.

[27] Note that the the area after the dilation and closing operation is actually the light gray **and** the dark gray area.

Throughout this paper we use a flat[28] circular structure element of varying radius.


(a) Morphological erosion (b) Morphological dilation

(c) Morphological opening (d) Morphological closing

Figure 6: The four basic morphological operators using a circular structure element. Black denotes the structure element, the red dot denotes the pixel for which we visualize structure element, dark gray denotes the image before the operation medium gray denotes the final image. For opening and closing the light gray area denotes the intermediate step.

In practice one comes a long way by just using these operators. There is however one more, *morphological reconstruction*, which is quite interesting and worth noting. Morphological reconstruction uses two images and a structure element, the *marker* and the *mask*, as opposed to just one image and a structure element. As described in [13] the reconstruction operator works in such a way that the peak of the marker image specify where the processing begins, the peaks are then dilated while being forced to fit within the bounds of the mask image. This process is then repeated until there is no more change in the marker image. The algorithm is as follows:

[28]When working with gray scale images one can use a non-flat structure element e.g. normal shaped. A flat is top-hat shaped.

```
**Algorithm** $Reconstruct(G, F)$
(∗ Reconstruction ∗)
**Input:** Marker Image $G$;
    Mask Image $F$
**Output:** Reconstructed Image $I$
1.  $h_1 \leftarrow F$
2.  **while** $h_{k+1} \neq h_k$
3.      $h_{k+1} \leftarrow (h_k \oplus S) \cap G$
4.  **return** $h_k$
```

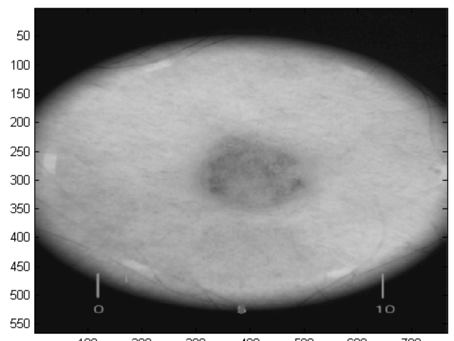
Figure 7: Algorithm for morphological reconstruction

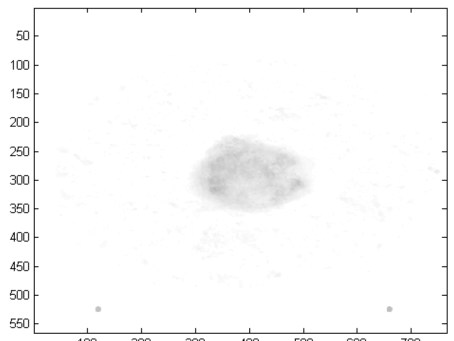
4.1.2 Morphological reconstruction to simplify segmentation

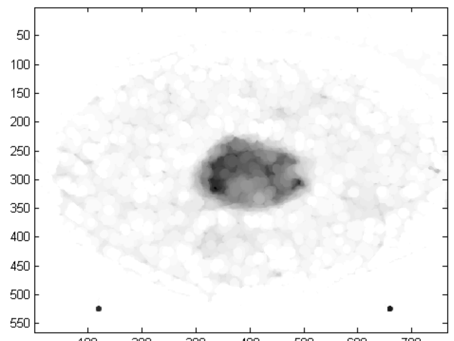
This is a method inspired from [10] since this makes the image seem more pronounced and should therefore help thresholding. We do this by reconstructing the image within the original image, I, as a mask and the opened image, I_o, as a marker, resulting in I_r. We now open the inverted image of I_r, call it $I^c_{r_o}$, and do another reconstruction using $I^c_{r_o}$ as a marker and the inverted image of I_r as a mask, resulting in I^c_{final}. The result is then simply the inverted image of I^c_{final}. The algorithm is shown in Figure 8 and visualized in Figure 9. The majority of the images we use in this paper were taken in a way such that we have a circular image with a black border. At a first glance, this seems easily solved, however making a general algorithm was not as trivial as expected. The most straight forward way to go, would be to simply crop the image, but to be able to do that we need information of the location of the lesion. To get this information we would need to, e.g. threshold the image, and we know that this is not possible since the border is dark, as is the lesion. There are some ways to overcome this problem such as figuring out the radius of the circle and use of some cloning for the areas outside this radius. This seemed to be the best option, although it would mean a lot of unnecessary work for such a simple problem. Luckily enough we found yet another morphological operator, *clear to border*, which is basically a special case of reconstruction. In short terms this clones low values that are connected to the border. Since the raw data is in RGB-color space, we know that visually dark areas are represented by low values over all channels which allows us to clear each channel with no worries, which might not have been possible in other color spaces such as LAB.

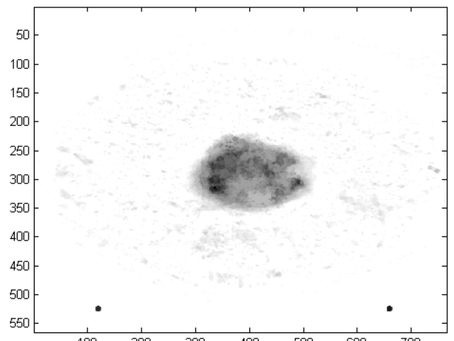
```
**Algorithm** $PreprocSegmentation(I)$
(∗ Preprocessing via reconstruction ∗)
**Input:** Image $I$
**Output:** Image $I$
1.  $I_o \leftarrow \text{Open}(I)$
2.  $I_r \leftarrow \text{Reconstruct}(I, I_o)$
3.  $I^c_{r_o} \leftarrow \text{Open}(255 - I_r)$
4.  $I_{final} \leftarrow 255 - \text{Reconstruct}(I^c_{r_o}, 255 - I_r)$
5.  **return** $I_{final}$
```

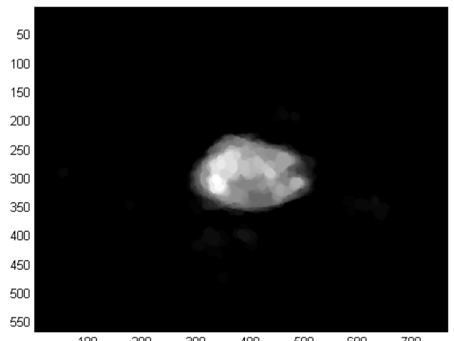
Figure 8: Algorithm for preprocessing the image with morphological reconstruction.


(a) Original image.


(b) Clear border; get rid of the dark border.


(c) Open; removes fine grained (light) noise.


(d) Reconstruct; use previous opened image as mask. Lesion is now more distinct.


(e) Open inverted image; removes fine grained (dark) noise.

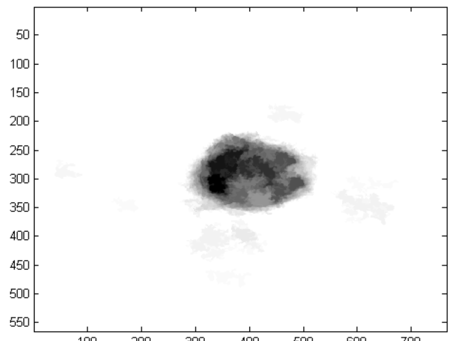

(f) Reconstruct and invert back; we now have a very distinct lesion.

Figure 9: The segmentation preprocessing step by step. As we can see we went from a fairly uncooperative image to something that's very easy to use for isolating the lesion.

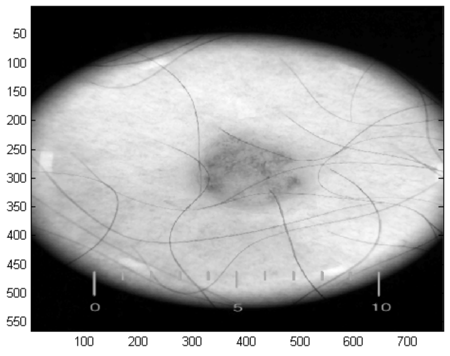
4.1.3 Hair removal

The method used to filter away hairs was inspired by [12], and is based on that hairs should absorb more light than skin and therefore should appear darker when looking at a luminosity gray scale image. Since luminosity is encoded in all channels in an RGB color space, we convert it to LAB color space, where L will represent luminosity. Since this channel contains only the data we are interested in we consider it to have the desirable range and quality. From this we could do a morphological opening to remove thin details, i.e. hair strands. Subtracting the original image from the opened image will indicate a higher difference where the hairs are present and from that we can do a simple thresholding to acquire a mask for the hairs. Now simply replace the masked regions in the original image with the opening. The algorithm presented in Figure 10.

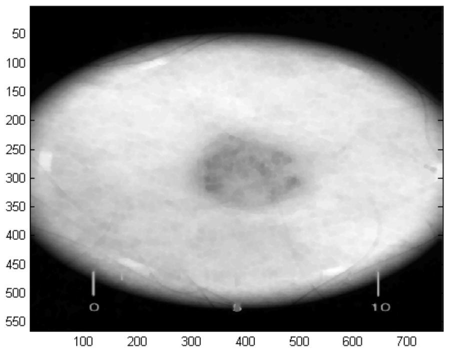
```
Algorithm RemoveHair($I$)
(* Remove Hairs *)
Input: Hairy Image $I$ in LAB color space
Output: Clean Image $I$ in LAB color space
1.  $I_o \leftarrow$ Open($I$)
2.  $M \leftarrow |I_L - I_{o_L}|$
3.  $M_b \leftarrow$ threshold($M$)
4.  for every index i,j
5.          if $M_b(i,j) = 0$
6.                  $I(i,j) \leftarrow I(i,j)$
7.          else
8.                  $I(i,j) \leftarrow I_o(i,j)$
9.  return I
```

Figure 10: Algorithm for removing hairs (I is all three channels. Writing e.g. I_L or I_{o_L}, means the luminance channel of that image).

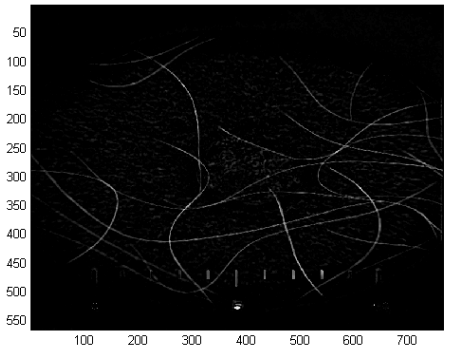
Note that this algorithm is not flawless since a worst case scenario appears when the hairs are darker than the skin but brighter than the lesion will cause the algorithm to fail. The simplest solution is to basically require that hair is physically removed before taking the picture, leaving this step completely unnecessary.



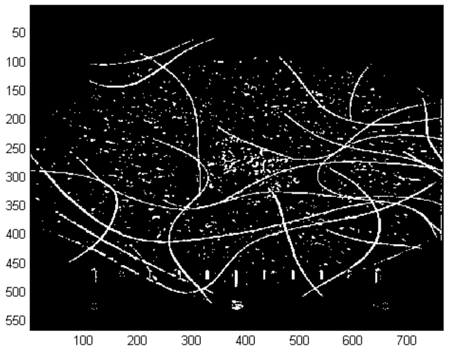
(a) Original image.



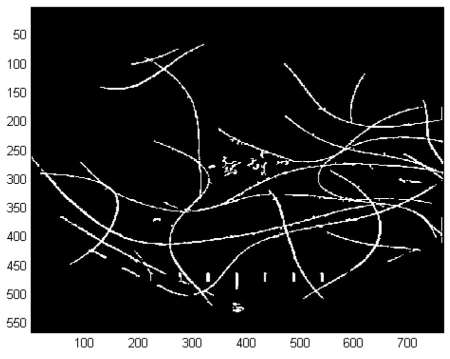
(b) Opened image.



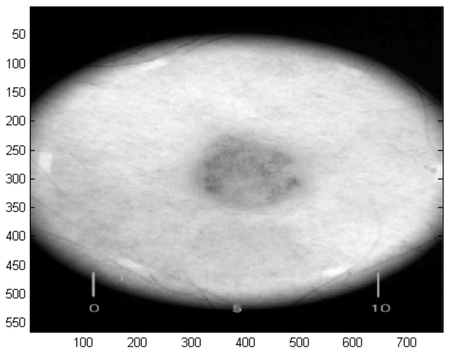
(c) Difference between (a) and (b).



(d) Thresholding.

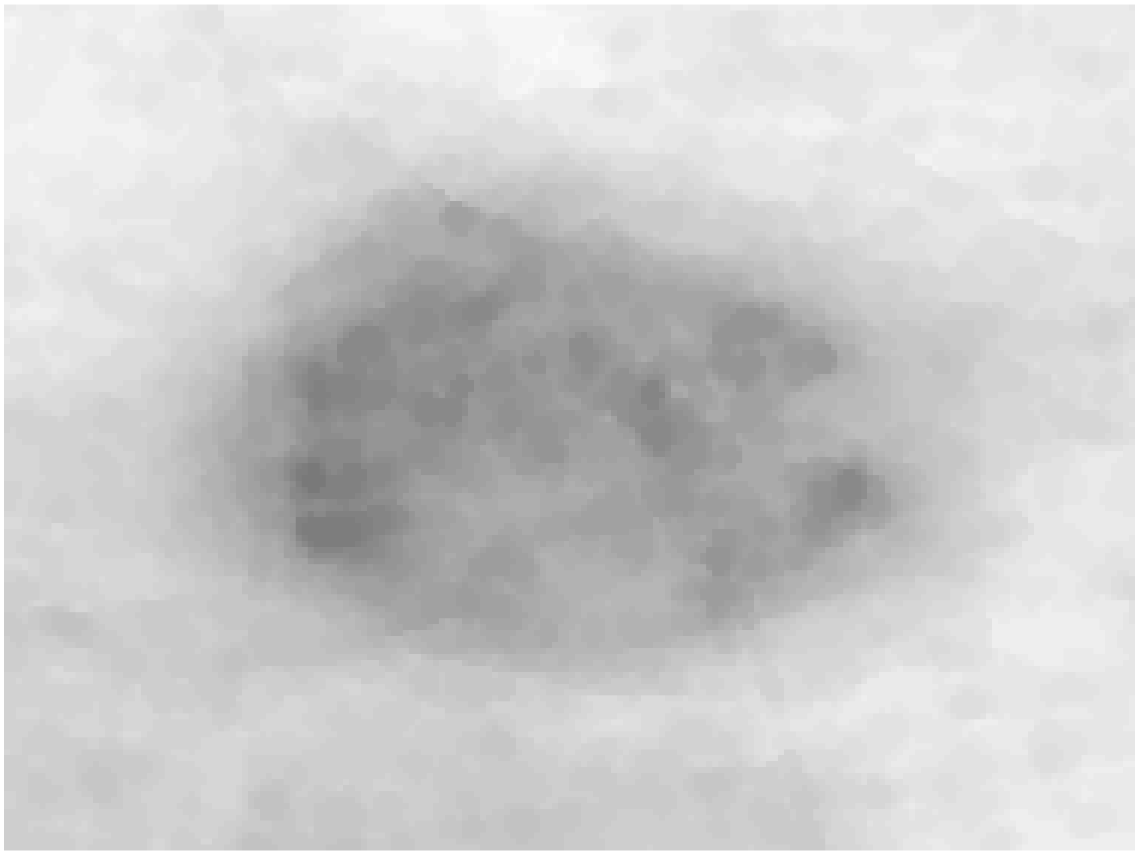


(e) Remove small areas



(f) Final image where the areas in the mask (e) have been replaced by (b), effectively removing the hairs.

Figure 11: Hair removal. The opened (b) image and the final image (f) seem identical. However, when we compare these two more closely (in Figure 12), we observe that using just opening, detail is lost.


(a) Opened image, note the circular smudges as a result of the opening.

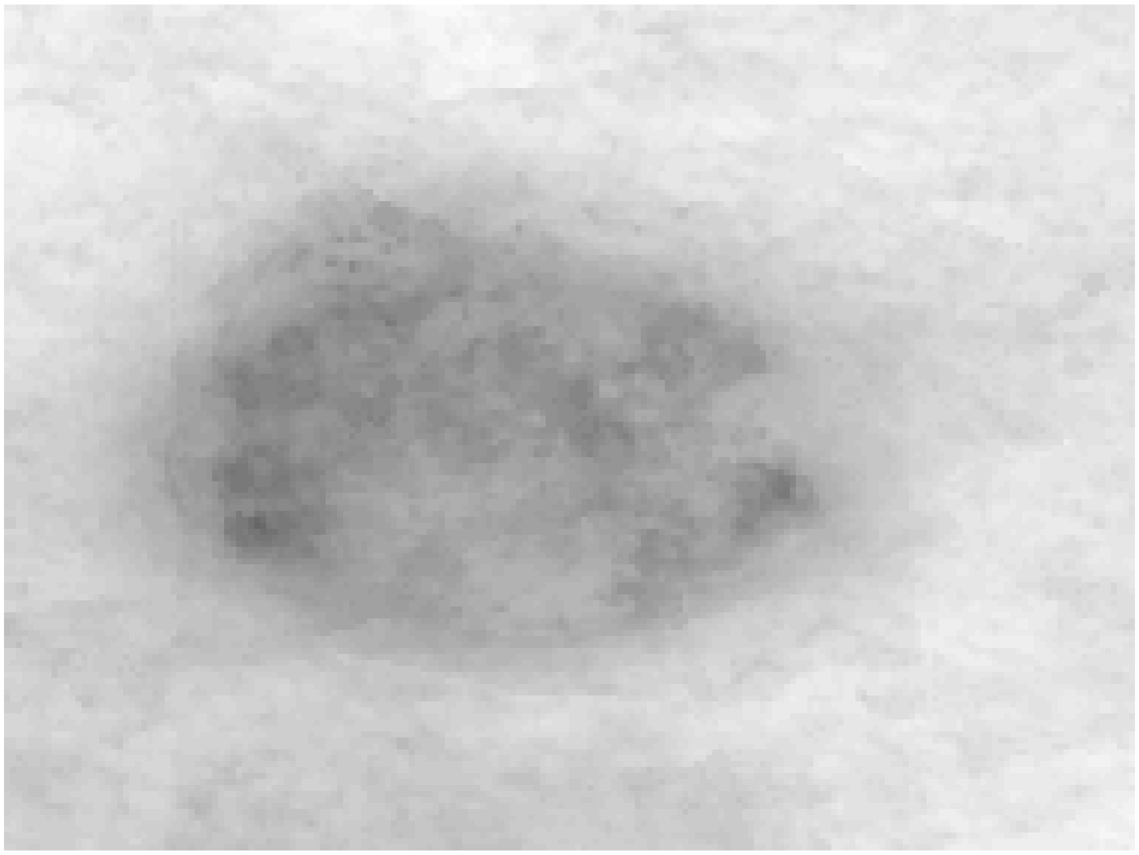

(b) Final image

Figure 12: Comparing these two images we observe that a lot of fine detail has been lost in the opened image (similar to a blur). As this step only aims to remove hairs we only replace the area of the hairs with the opened image. This way we remove hairs with minimal alterations to the original image.

4.1.4 Normalize colors

Depending on the skin color or lighting conditions on the subject, the lesion may be biased towards different colors. Given e.g, a true color 24-bit image (i.e. where the red green and blue channel can be represented integer values from 0 to 255), we can compensate for the lighting by subtracting the mean color of the entire image, and re-normalize to fall in the range 0-255, we do this in the following way:

1. Let R, G and B represent the red, green and blue channels of the image, represented as matrices, where the values range from 0 to 255.
2. Let $\bar{r}$, $\bar{g}$ and $\bar{b}$ be the mean values of all the elements in the corresponding matrices.
3. Define new channel values as $R_n = R - \bar{r}$, $G_n = G - \bar{g}$ and $B_n = B - \bar{b}$.
4. Also define M as the single largest element from R, G or B, as well as m being the smallest.
5. Finally let I_n be the image composed of the new channels and normalize I_n by $I_n = 255\frac{I_n - m}{M - m}$.

4.1.5 Noise removal

Noise removal is a pretty common technique used in image processing. There are different ways to do this [8], the simplest probably being a median filter [2]. The idea is to, given a neighborhood of a pixel p; we simply replace the value of p with the median of the neighborhood. Depending on how heavy you want your noise removal you can choose larger neighborhoods and/or iterate the method on itself. There certainly are more fancy methods as well, such as using wavelets or Fourier transforms. The idea is basically that since we know that noise will be of high frequency and carries little energy, we simply dampen the corresponding constants of small amplitude before we do our reconstruction. The most common way to modify the constants is commonly called *hard thresholding* and *soft thresholding*.

4.2 Thresholding

Thresholding is the core of many algorithms related to image processing, since it helps us to distinguish between the data we need and the data we do not need. A histogram is an estimate of the distribution of values for a signal, generally represented as a number of bins containing the number of occurrences of values within different intervals. For image processing in general, and in this thesis, we use consecutive, non-overlapping intervals of equal length. When we want to threshold an image or a signal, we usually look at the histogram to find a minimum, which will indicate a desired thresholding limit. Depending on the setup, one may be looking for one or more minimas, and thus need to change the thresholding algorithm correspondingly. However in our case we only want to mask the lesion, and thus we try to find a histogram with only one local minimum. To get a suitable histogram we use the L-channel from the LAB-color space, since this channel will generate a histogram with two peaks; one for the dark areas, i.e. the lesion, and one for bright areas, i.e. the skin.

A straight forward approach to find the minimum would be to start at the smallest value in the histogram and successfully take one step forward as long as the histogram increase in value. When the histogram no longer increase value, we know that we are at a local maximum, and continue stepping forward, now as long as the histogram decrease in value. When it no longer decreases, we know that we are at the minimum. It is easy to realize that this approach have one fatal flaw; spikes in data can cause the algorithm to stop too early. So if we would like to use this algorithm properly we should, as in noise removal, e.g. use some median filter to smoothen the histogram, which has to be fine tuned to handle the trade off in quality versus stability.

When we deal with natural signal, another approach, called **minimum error**, is to assume the histogram to be a sum of two or more normal distributions. To find the local minimas, we then simply fit the distributions to the data and have our minimum as the intersections of the distributions.

The third option, which is also used in this paper is the midpoint method which is commonly defined as in the algorithm in Figure 13. This algorithm finds the threshold value such that the mean value for the part of the histogram below the threshold equals the mean value of the part of the histogram above the threshold.

```
Algorithm threshold($I, T_0, T_1$)
(* Threshold, midpoint method *)
Input: Grayscale image $I$
Input: Arbitrary thresholds $T_0 \neq T_1$
Output: Threshold $T_n$
1.  $h \leftarrow histogram(I)$
2.  while $T_{n+1} \neq T_n$
3.          $\mu_0(T_n) \leftarrow (\sum_{z \leq T_n} zh(z))/(\sum_{z \leq T_n} h(z))$
4.          $\mu_1(T_n) \leftarrow (\sum_{z > T_n} zh(z))/(\sum_{z > T_n} h(z))$
5.          $T_{n+1} \leftarrow \frac{\mu_0(T_n)+\mu_1(T_n)}{2}$
6.  return $T_n$
```

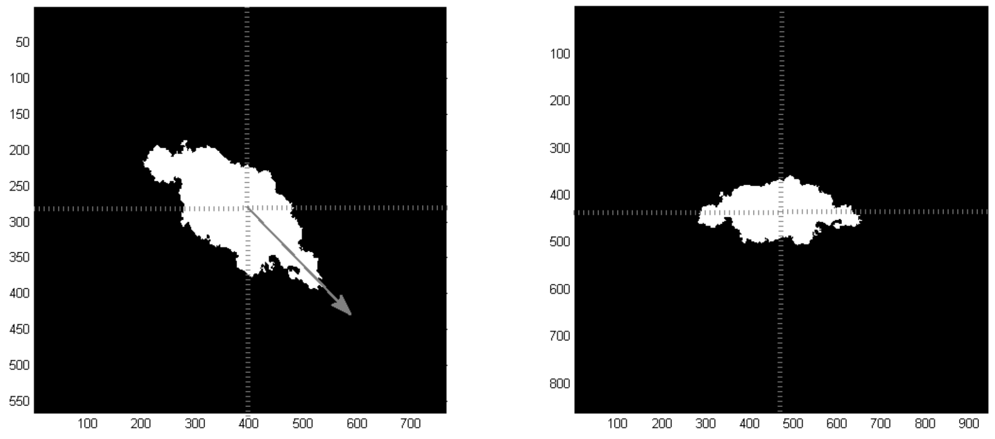
Figure 13: Algorithm for thresholding. Here z is an intensity and $h(z)$ is the histogram value at z, and since the histogram is discrete, z represents an index to the histogram.

4.3 Rotate and crop the image

By using the positions of the masked pixels from the binary thresholded image, we can preform a PCA to get a rotation such that the rotated image will be optimally aligned to the axes such that each segment of the different quadrants will be as similar as possible. This is useful in our case since we can effectively crop the image and also investigate asymmetry in order to analyze e.g. the variance of the means and the variance of the variance between the different quadrants. The procedure is as follows:

1. Given the masked pixels; store the corresponding positions in a $(2 \times n)$ matrix, X, where n is the number of masked pixels.
2. Let: $(U, S, V) = SVD(X)$
3. Denote the first column of U as $b = (b_x, b_y)^T$, and let $a = (1, 0)^T$. Now a is our origin vector and b is the principal direction.

4. To get the angle, θ we use some simple trigonometry[29]: $\theta = sign(b_y)cos^{-1}(b_x)$.
5. Using e.g. MATLAB, we can now rotate the image, I, by $I = imrotate(I, \theta)$



(a) Original.

(b) Image after rotation.

Figure 14: Segmentation preprocessing. In image (a) the arrow marks first principal component. In the rotated image (b) each segment now has roughly equal coverage

Since we now have a rotated image which fills an axis aligned bounding box nearly optimally, we can easily crop the image by just finding the maximum and minimum values of the indexes corresponding to the white pixels. When we use our cropped image we can save computations and data, and as we will see later this will also make it easier for us to disregard some redundant data, providing us better parameters.

[29]Making sure that b is normalized.

4.4 Feature Extraction

To recap, the preprocessing procedure for the images so far, is performed via these steps in the following procedural algorithm:

Step 1: Clear border;

Step 2: Remove noise;

Step 3: Remove hairs;

Step 4: Reconstruct;

Step 5: Threshold;

Step 6: Rotate and crop image.

4.4.1 Skewness and kurtosis

Mean and variance are probably the most used statistical parameters, as well as self explained. In many cases these are good enough, but there are a couple of more instruments in our toolbox, namely *skewness* and *kurtosis*. Skewness describe how centered the distribution is around the mean, a skewness of 0 means that the distribution is perfectly centered around the mean. Kurtosis describe the tails of the distribution. The normal distribution has a kurtosis of 0, negative values indicate lighter tail and positive values indicate heavier tails. Below we describe the different parameters.

Mean:

$$E[x] = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Variance:

$$E^2[x] = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2.$$

Skewness:

$$E^3[x] = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^3}{\sqrt{(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2)^3}}.$$

Kurtosis:

$$E^4[x] = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^4}{(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2)^2} - 3.$$

Here, N is the number of samples, x_i the i^{th} sample, and $\bar{x}$ the mean of all samples i.e. $E[x]$.

4.4.2 Wavelet features

In [7] was shown that wavelets were created in the early 1980's as a response to a debate between whether the visual system react to spatial frequency rather than space. In this wavelet model of vision, what we see as an image is the signal, the receptive fields correspond to the wavelets and the constants are the reaction of a neuron to a particular pattern of light. Small receptive fields would tell us about space whereas large would tell us about frequency. In alignment with this there are theories that state that our visual system uses what is similar to a wavelet transform, which would help us differentiate different objects. Thus some believe that the future of wavelets is not necessarily in compression, but rather recognition. Further, [7] states that there are infinitely many wavelets and that there is no real guideline on which wavelets to choose [30]. Thus in this paper mainly common wavelets with a small number of vanishing moments, such as *Haar, Daubecies2* and, *Daubecies4* were investigated which are visualized along with their filters in Figure 15.

[30]Except for numerical analysis where vanishing moments does have a great impact.

Haar is the simplest wavelet to define with:

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2}, \\ -1 & \frac{1}{2} \le t < 1, \\ 0 & \text{else.} \end{cases} \quad \text{and} \quad \phi(t) = \begin{cases} 1 & 0 \le t \le 1, \\ 0 & \text{else.} \end{cases}$$

Here, ψ and ϕ is the mother wavelet and scaling function, respectively.
The Daubecies wavelets on the other hand is a polynomial generalization of the haar wavelet where *Daubechies1* equals Haar. The number indicates the degree of the polynomial as well as the vanishing moments, where Daubechies1 would have zero vanishing moments, Daubecies2 would have one, Daubechies4 three, etc. Except from Daubechies1 there is no explicit form for the wavelet functions of this family and they consequentially defined as a product filter in the frequency domain. We can however approximate the wavelets in order to visualize them with the *Cascade algorithm*. To reconstruct the mother wavelet we can start with a unit impulse at the coarsest level as successively reconstruct the signal to the desired level from where the scaling function also can be reconstructed. As can be expected from [6], besides using mean values and variance, skewness and kurtosis may help identifying malign tissues. The reasoning for this is that natural effects, such as healthy tissue, are often normally distributed in some sense and deviations from the normal, such as in microcalcifications, results in a deviation from this distribution. Hopefully, we will be able to identify similar deviations in this manner as well. Since both skewness and kurtosis can have negative values we include the absolute value in our model since a value of zero is assumed to be the indicator of healthy tissue, and thus deviations from this will indicate unhealthy tissue. In [10] was shown that using the mean and variance of the wavelet coefficients from the binary image as features for a neural network they were able to achieve good classification rate. Therefore we will include the same parameters in this paper as well. As in the paper [10] we will use approximately three levels of decomposition. Since we here deal with a binary image, we need not worry about wavelet extension, since a zero padding will fit the signal exactly.
The procedure is quite straight forward:

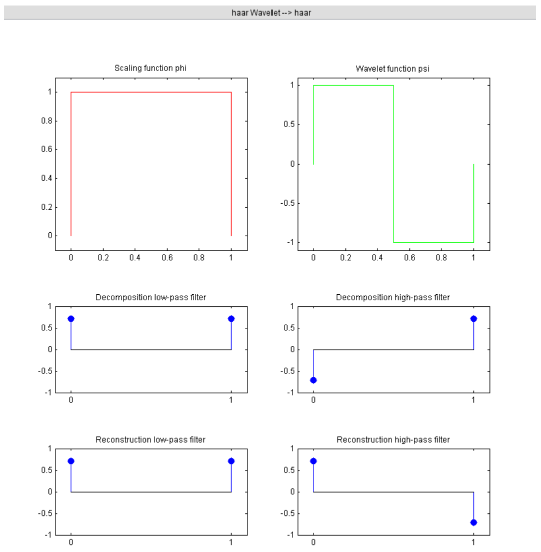
1. *Given a binary image, I.*

2. *Decompose the I for n levels, and store the horizontal, vertical and diagonal details in the matrices H_i, V_i and D_i, for the corresponding i:th level of decomposition.*

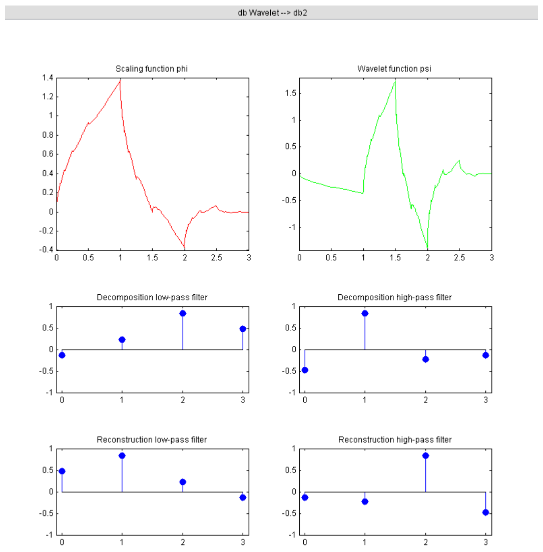
3. *Store the mean, variance, skewness and kurtosis for each detail and level as parameters, that is $12n$ number of parameters.*

To reassure us that these parameters have any significance at all, we do a small test with a couple of extreme cases to see if we notice any difference. For this setup we have two benign pictures shown on Figure 16-(a),(b) which seem to have a small color variation, and two malign pictures shown on Figure 16-(c),(d) with high variation. Both images have been rotated and cropped to keep redundant data outside the lesion to a minimum. The images in Figure 16 are then decomposed and we investigate the coefficients for the horizontal, vertical and diagonal details, for each row and column independently. During some tests in the beginning of the project it was noticed that, for wavelet coefficients, the resolution of the images could have greater impact on the parameters than the actual features, so to be consequent we rescale all the images to have equal resolution [31]. The spectrum plots in the following figures we have the coefficients at the x-axis and the corresponding value at the y-axis followed by tables summarizing the values. These plots and tables are meant to give us some intuition whether there is any difference between the two cases [32]. We also have a *ground truth* image which is composed of plain normal noise, as seen in Figure 17, which is meant to represent some ideal case as a reference.

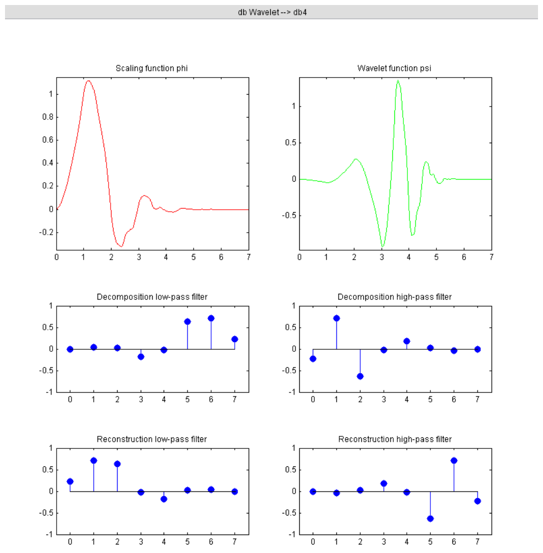
[31] This can be tricky as well since rescaling can involve some bi-linear or bi-cubic interpolation, which could be exposed because of vanishing moments.
[32] We will here write e.g. DB2 to denote the Daubechies2 wavelet.



(a) Haar (Daubechies1)

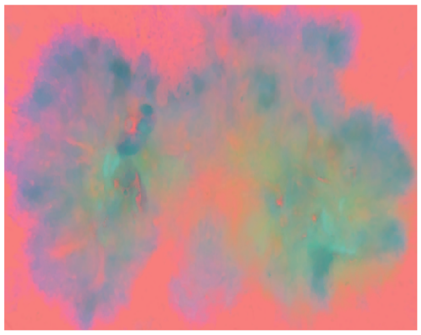


(b) Daubechies2

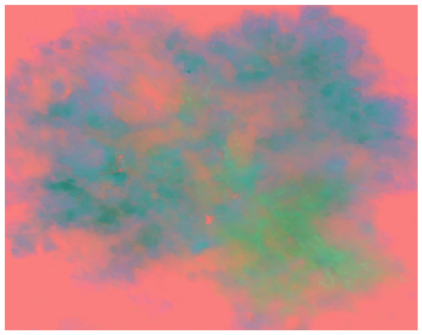


(c) Daubechies4

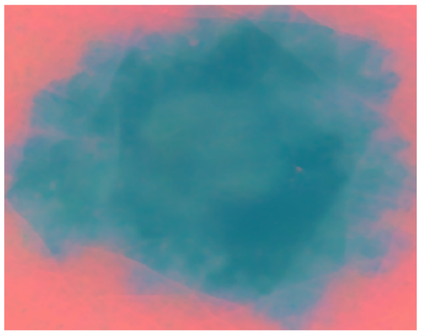
Figure 15: Examples of wavelets and their sample coefficients (filter).



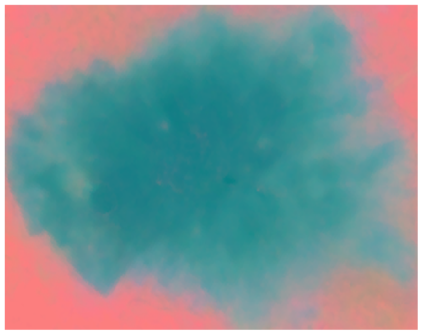
(a) Malign1.



(b) Malign2.

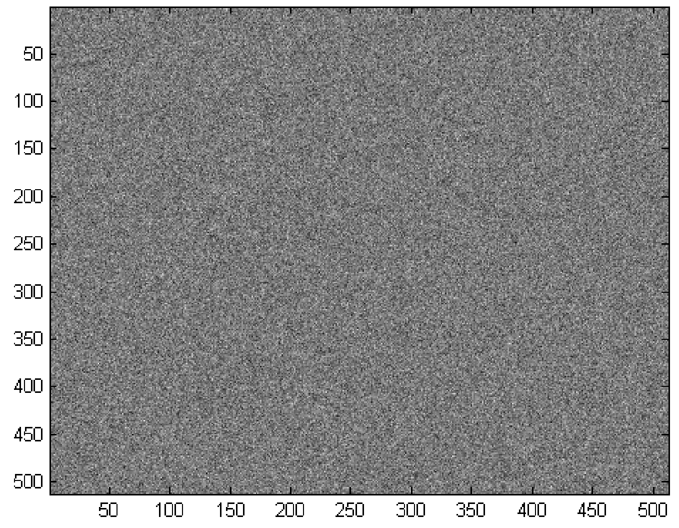


(c) Benign1.

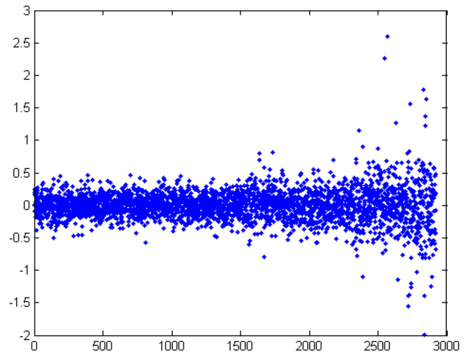


(d) Benign2.

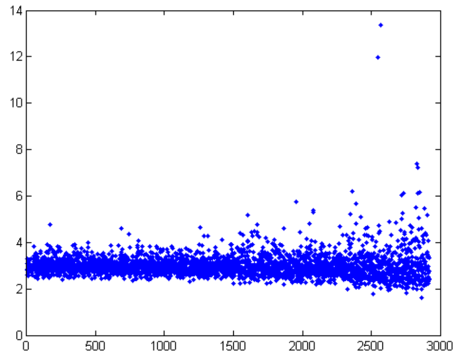
Figure 16: Cropped test images. Represented by mapping the LAB-values to RGB (explained in section 4).



(a) Normal noise.



(b) Skewness.

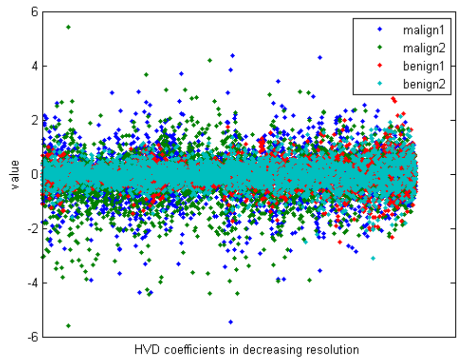


(c) Kurtosis.

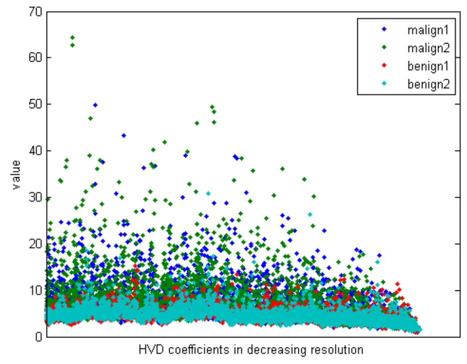
Figure 17: Ground truth of skewness and kurtosis values. Using a image of plain normal noise with mean 0 and variance 1. Using the DB2 wavelet.

	Normal
Skewness absolute mean	0.1990
Skewness variance	0.1023
Kurtosis mean	2.9854
Kurtosis variance	0.3149

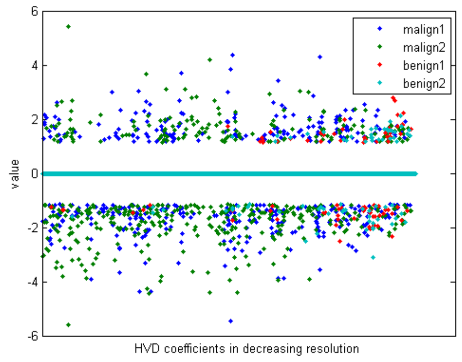
Table 1: Skewness and kurtosis values for the control image in Figure 17.



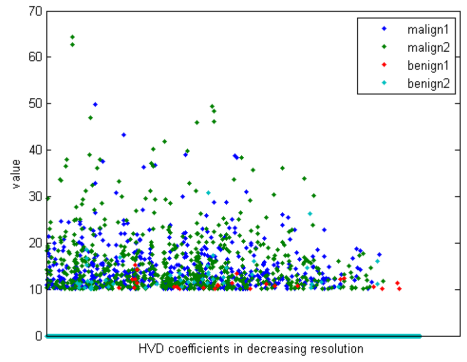
(a) Skewness.



(b) Kurtosis.



(c) Skewness - thresholded.



(d) Kurtosis - thresholded.

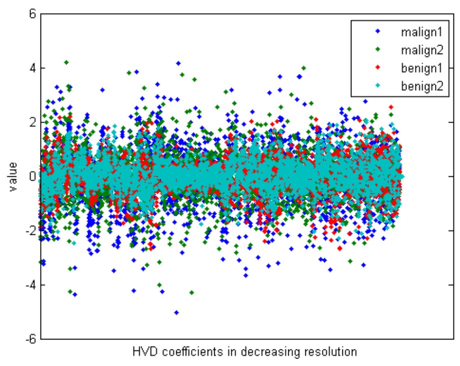
Figure 18: Spectrum of skewness and kurtosis for the DB4 wavelet. In (a)-(b) the full spectrum is plotted whereas in (c)-(d) values below a certain threshold are disregarded.

DB4	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	0.5014	0.7190	0.2972	0.3189
Skewness variance	0.5242	1.0891	0.1912	0.2171
Kurtosis mean	6.9720	8.5350	5.1325	4.9342
Kurtosis variance	16.5221	43.6926	4.8227	4.6182

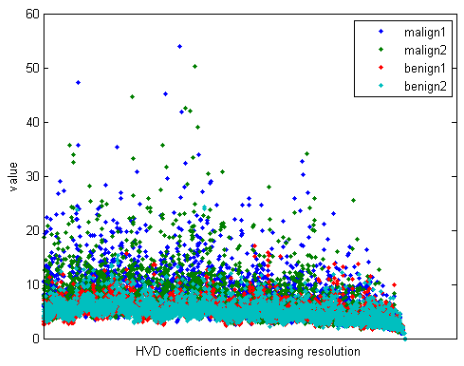
Table 2: Mean and variance of skewness and kurtosis from the spectrum in 18 (a)-(b).

DB4	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	1.8160	2.0639	1.5549	1.5725
Skewness variance	0.3633	0.8117	0.0648	0.0911
Number of values above threshold	428	493	41	32
Kurtosis mean	16.7860	19.4424	14.8717	13.0884
Kurtosis variance	38.3466	78.9864	12.4312	1.4258
Number of values above threshold	492	576	53	20

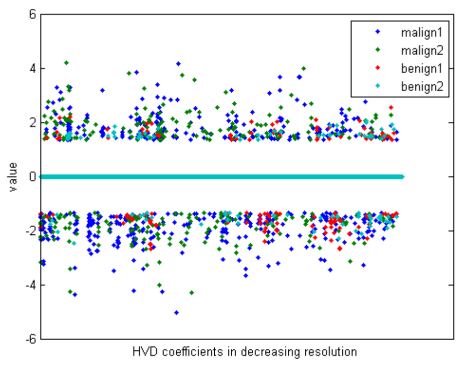
Table 3: Mean, variance and number of values above threshold for skewness and kurtosis from the spectrum in Figure 18 (c)-(d).



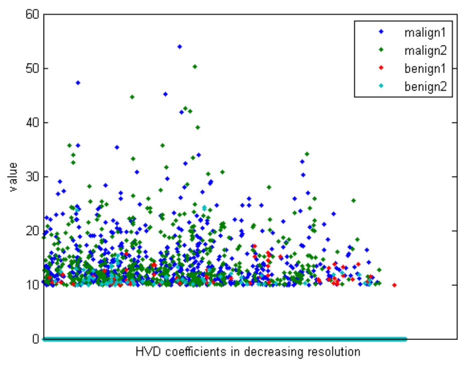
(a) Skewness



(b) Kurtosis



(c) Skewness - thresholded



(d) Kurtosis - thresholded

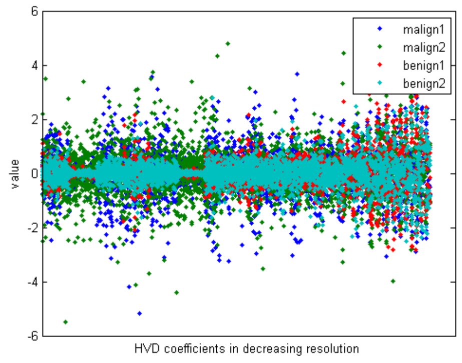
Figure 19: Spectrum of skewness and kurtosis for the HAAR wavelet. In (a)-(b) the full spectrum is plotted whereas in (c)-(d) values below a certain threshold are disregarded.

HAAR	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	0.7163	0.5999	0.4716	0.4410
Skewness variance	0.8823	0.6964	0.4062	0.3426
Kurtosis mean	7.4629	7.3068	5.6624	5.2897
Kurtosis variance	15.0520	23.3291	4.4196	3.2257

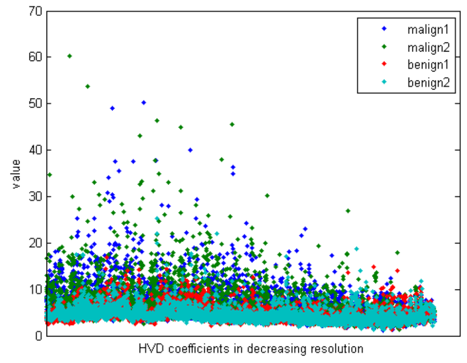
Table 4: Mean and variance of skewness and kurtosis from the spectrum in Figure 19 (a)-(b).

HAAR	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	1.8482	1.9269	1.6742	1.6170
Skewness variance	0.2543	0.3660	0.0958	0.0673
Number of values above threshold	448	296	147	104
Kurtosis mean	13.7607	15.2561	12.2406	11.5498
Kurtosis variance	19.4261	43.0680	7.2712	4.2840
Number of values above threshold	523	495	107	56

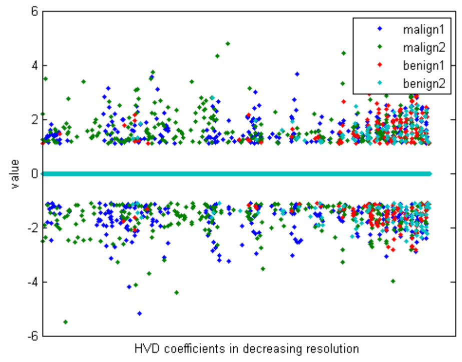
Table 5: Mean, variance and number of values above threshold for skewness and kurtosis from the spectrum in Figure 19 (c)-(d).



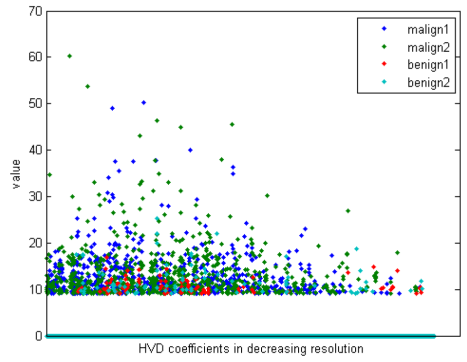
(a) Skewness



(b) Kurtosis



(c) Skewness - thresholded



(d) Kurtosis - thresholded

Figure 20: Spectrum of skewness and kurtosis for the DB8 wavelet. In (a)-(b) the full spectrum is plotted whereas in (c)-(d) values below a certain threshold are disregarded.

DB8	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	0.5564	0.5467	0.2303	0.2275
Skewness variance	0.6069	0.6177	0.1095	0.1206
Kurtosis mean	7.7029	7.1874	4.5109	4.5519
Kurtosis variance	17.7679	21.7402	2.6856	2.6301

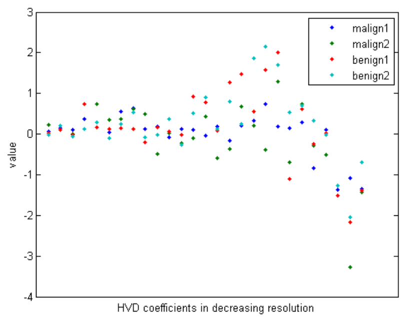
Table 6: Mean and variance of skewness and kurtosis from the spectrum in Figure 20 (a)-(b).

DB8	Malign1	Malign2	Benign1	Benign2
Skewness absolute mean	1.5681	1.6417	1.2838	1.4467
Skewness variance	0.3009	0.4448	0.0762	0.2790
Number of values above threshold	505	447	53	50
Kurtosis mean	14.0525	14.8454	11.9737	11.9704
Kurtosis variance	24.5558	41.8609	3.7044	13.9579
Number of values above threshold	647	537	47	39

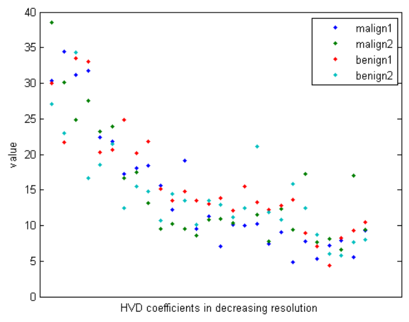
Table 7: Mean, variance and number of values above threshold for skewness and kurtosis from the spectrum in Figure 20 (c)-(d).

We see from Figures 17-20 and Tables 1-7 that when we use the cropped image there is indeed a notable difference between the two cases. All of the values from the malign lesions are higher than for the benign with some promising candidates such as number of values above threshold, kurtosis variance and skewness variance. Compared to the ground truth in Figure 17 and Table 1 we consider the difference between the parameters to be reasonable as well. As expected we got the most efficient results from the Daubechies2-Daubechies4 wavelets. Haar and higher order Daubechies either had worse results or no significant improvement. This was to be expected since as discussed earlier, higher order wavelet was not particularly useful for image analysis. We also note from Figures 17-20 that the majority of the values are centered around the mean. This is expected since in natural scenes, the most information is obtained at the low frequencies [7]. This introduces a bit of a complication, since if we would like to take the mean of all the values, it would be tough to see a difference. To compensate for this we simply disregard values close to the mean by setting them as zero when, $|\beta| < \overline{|\beta|} + \hat{\beta}$, where $\overline{|\beta|}$ and $\hat{\beta}$ are the mean of the absolute values and standard deviation of the values respectively. The last observation is that we get nonsense at the coarsest scale, especially for higher order wavelets such as Daubechies8. This is also to be expected since in this analysis we decompose the signal to the coarsest scale possible. Recalling that we need to do extensions for border cases, we realize that when we try to decompose at the coarsest level the majority of the information will come from the extension and not the actual signal.

It was also realized that, since the normal distribution is separable, it would make sense to just investigate the skewness and kurtosis on the whole matrix, instead of the rows and columns independently. However this provided poor results, as can be seen in table 8 and figure 21, and thus we stick to the row-column approach.



(a) Skewness.



(b) Kurtosis.

Figure 21: Using data from whole matrix with the DB2 wavelet.

DB2	Malign1	Malign2	Benign1	Benign2
Skewness mean	0.3587	0.6206	0.6521	0.5994
Skewness variance	0.2867	0.8026	0.8709	0.7197
Kurtosis mean	14.6403	15.1114	16.2085	14.4795
Kurtosis variance	79.0416	65.0885	56.7736	41.9747

Table 8: Mean and variance of skewness and kurtosis from Figure 21

.

Knowing this we have our following algorithm for calculation of the parameters:

1. *Given a multi channel image*[33] *I.*
2. *Decompose the I for n levels, and store the horizontal, vertical and diagonal details in the matrices H_i^c, V_i^c and D_i^c. for the corresponding level i of the decomposition and channel c.*
3. *Calculate the skewness and kurtosis for every row and every column for every matrix, and store all the values in one long vector, v.*
4. *Set $v_i = 0$ if $|v_i| < \overline{|v|} + \tilde{v}$, where $\overline{|v|}$ is the mean of the absolute values and $\tilde{v}$ is the standard deviation.*
5. *Store the mean and variance of v as parameters.*

[33]Here we decompose each channel separately

4.4.3 Geometric features

According to [1], for a binary image, given N as the perimeter and A as the area both measured in number of pixels, we can define our irregularity index as

$$Ir := \frac{N^2}{4\pi A},$$

Other parameters that may also be of value can be defined as, solidity:

$$So := \frac{A}{C_A},$$

and convexity:

$$Co := \frac{C_N}{N},$$

where C_A is the area of the convex hull and C_N the convex perimeter. We do not focus much on pure geometric features in this paper and thus these simple definitions have to suffice.

4.4.4 PCA for neighbors

In [3], a technique to do a spatio-chromatic analysis of images using principal components is proposed. That is, instead of doing the PCA per pixel, the neighborhood of the pixel is used as input data. In this paper we use a (3×3) pixel neighborhood. As was discussed with wavelets, this was also created with the visual receptors in mind, where the neighborhood would correspond to the receptors in an eye or a camera. Not enough time were spent investigating this, so for this paper we simply calculate the mean and variance of the different color channels of the bases. If one were to take this further it would make sense to investigate the average case for the bases of benign lesions and malign lesions respectively. Hopefully one would be able to notice a difference in terms of the basis. Another interesting aspect is that when we look at the projections onto the different bases, we could hopefully catch details that would otherwise not be that apparent. In Figures 23 and 24 we show the bases and some of the projections onto these bases, respectively, using the image in Figure 22.

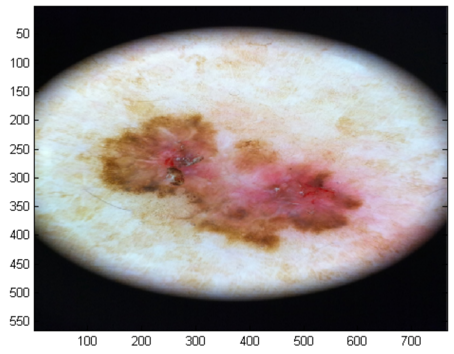
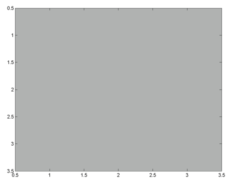
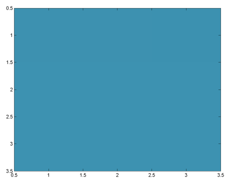
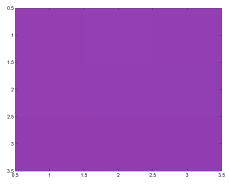
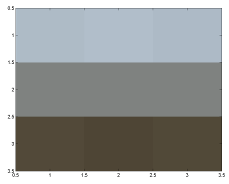


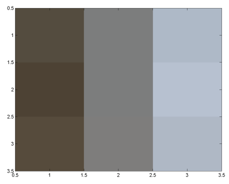
Figure 22: Original image

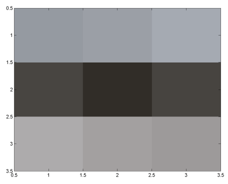

(23.1) e_0

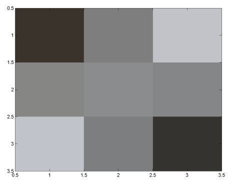

(23.2) e_1

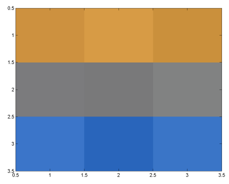

(23.3) e_2

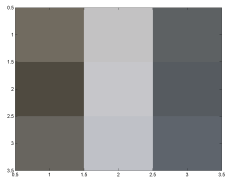

(23.4) e_3

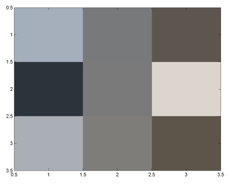

(23.5) e_4

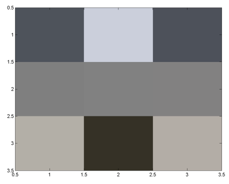

(23.6) e_5

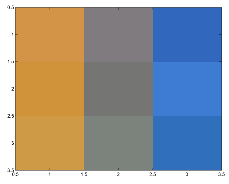

(23.7) e_6

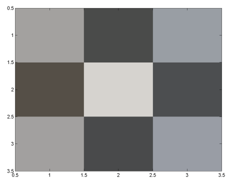

(23.8) e_7

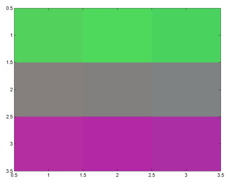

(23.9) e_8

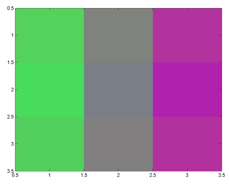

(23.10) e_9

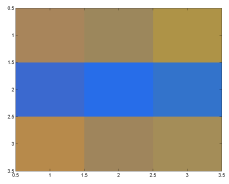

(23.11) e_{10}

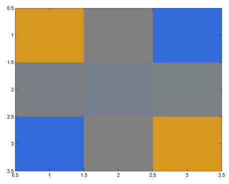

(23.12) e_{11}

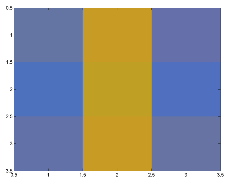

(23.13) e_{12}

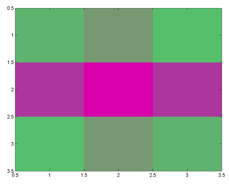

(23.14) e_{13}

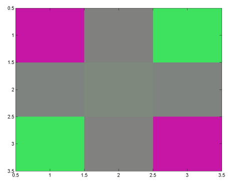

(23.15) e_{14}

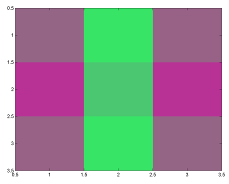

(23.16) e_{15}

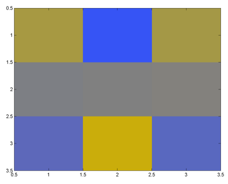

(23.17) e_{16}

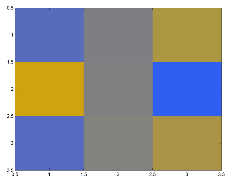

(23.18) e_{17}

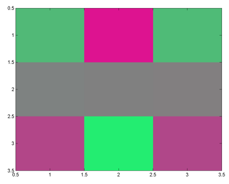

(23.19) e_{18}

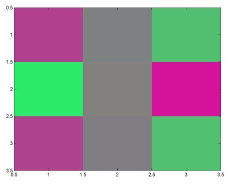

(23.20) e_{19}

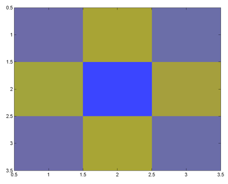

(23.21) e_{20}


(23.22) e_{21}


(23.23) e_{22}


(23.24) e_{23}


(23.25) e_{24}


(23.26) e_{25}

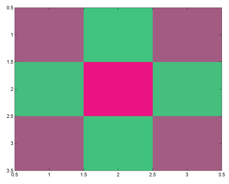
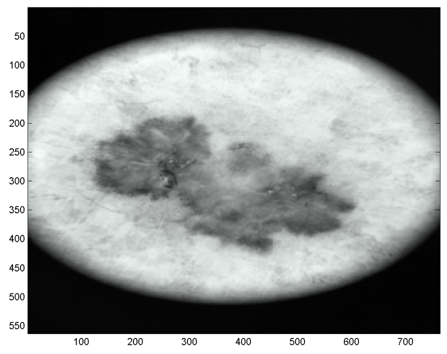
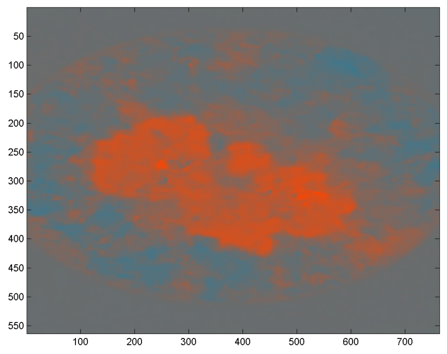

(23.27) e_{26}

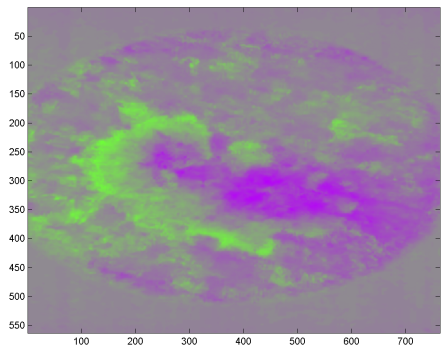
Figure 23: The different bases, where e_n is the n^{th} base, obtained for the described method. For clarity the 27 values are visualized as a (3×3) area with an RGB value for each sub area.



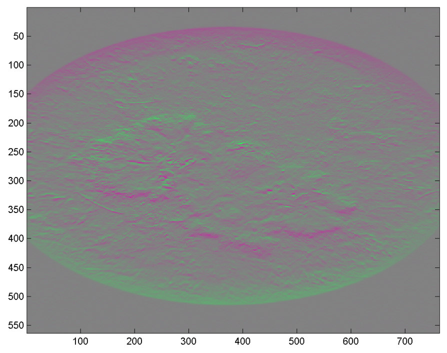
(24.1) Projected on e_0



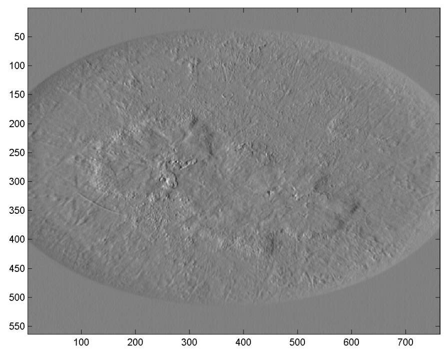
(24.2) Projected on e_1



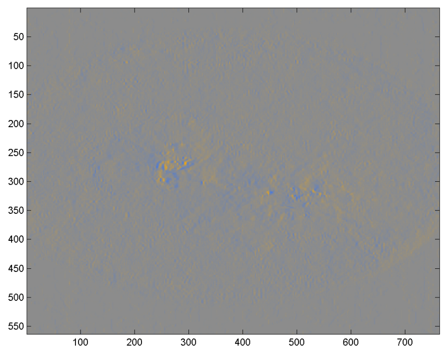
(24.3) Projected on e_2



(24.4) Projected on e_3



(24.5) Projected on e_4



(24.6) Projected on e_{14}

Figure 24: The image in Figure 22 reprojected onto the respective base. Note that different features are more prominent in different projections.

4.5 Evaluating the model

To evaluate the model a simple stepwise linear regression was used. The p-values used in these algorithms is a measure of the evidence against our null hypothesis, in this case the null hypothesis is that there is no relationship between the model and the data. The lower the p-value the stronger evidence against the null hypothesis. For stepwise regression there are generally three types of algorithms, namely:

1. **Forward selection** - Start with no parameters and successively add the most relevant feature, i.e. the one which has the lowest p-value when included. We stop after a certain number of parameters or when a certain threshold of the next p-value is violated.
2. **Backward selection** - Start with all the parameters and successively remove the one with the highest p-value until criterion is met.
3. **Forward and Backward selection** - Start with some model and alternate between removing and adding parameters.

In this paper the forward and backward selection, including cross product terms with no parameters as starting model was used. The reason for this is that since we have so many parameters that we don't really know will be significant or not, so we basically let the algorithm decide what's the best fit. The reason for allowing cross product terms is that when doing the analysis in practice the more different indications you have can be as dangerous as if you have one really strong indication, e.g. if you have a small color variation as well as an irregular border, this can indicate as much as if you had just a really strong color variation. Hopefully the cross products will capture these dependencies. We also assume that we will need many parameters in our model since, again in practice, you can not just look at one feature but rather many. Even though the **ABCDE-model** seems to have five features, every letter has many sub-indicators. Since we will allow so many parameters in our model, we have a risk of over fitting, so one should not trust the results completely.

To make things even more complicated there are many sub-classes of malign and benign melanoma, each with different characteristics. To investigate every possible class is beyond the scope of this paper, and we are thus as bold to generalize to a binary model, either it is benign or it is not. We thus expect our model to look as follows:

$$y = \beta_1 + \beta_2 + \cdots + \beta_1\beta_2 + \beta_1\beta_3 + \cdots + \beta_2\beta_3 + \ldots.$$

Where β_i is the i:th feature. If y is greater than a value, α, we declare the lesion malign. The value of α is can be calculated as the value that maximizes sensitivity[34] and specificity[35]. This means that by tweaking α we can get e.g. better sensitivity at the cost of specificity, which can be a worthwhile trade off.

When we have acquired our model we use cross validation to investigate the quality of the model which is done by these steps:

1. *Use all the data to figure out what parameters to include.*
2. *Take e.g. 70% as training data and 30% as test.*
3. *Fit the training data to the model acquired in step one using simple regression.*
4. *Evaluate the test data and save results.*
5. *Go to step 2. Stop after n steps or until convergence of e.g. the mean squared error of the model or the misclassification rate.*

[34]Sensitivity - Percentage of correctly identified malign lesions.
[35]Specificity - Percentage of correctly identified benign lesions.

5 Results and discussion

The main result of this work is that we fairly often managed[36] to get a sensitivity and specificity around 80%, with some best results at 93%. It should however be noted that the models which was acquired had around 10-15 parameters, which can be considered relatively high compared to our dataset with a size just over a hundred. The model is thus unfortunately not good enough to be used in practice as it would require consistent results with well above 90% in sensitivity which, with the reasons above, is something we can't guarantee.

As the saying that a chain only is a as strong as its weakest link, there is a similar problem with this work. For example, one of the most crucial parts is the thresholding and segmentation since all other parameters in one way or another, depends on the quality of the segmentation. It wouldn't matter if we had the best algorithms in the world for extracting parameters if they were extracted from the wrong part of the image. And, even though the thresholding process can be simple enough in its own right it also heavily depend on how we preprocess the image. To further illustrate the importance of spending more time on each small step; Tim K. Lee, David I. McLean, and M. Stella Atkins devote a whole paper [15] on just defining a measure for the boarder of the lesion[37].

In this paper we used a linear model to classify two different sets of data, which seem to work relatively well although it is a crude approximation. To achieve a better classification ratio we should use some logistic model, or even better support vector machines or neural networks. We also do not go more into specific details about exactly which parameters that were selected in our model, but it is however worth mentioning that there were, more or less, a consistency in the parameters chosen and that they were both depending on the skewness, kurtosis as well as the mean and variance of the binary images. We thus conclude that although it is hard to find single to few parameters describing the lesion, we have seen that it does however seem possible to come up with some kind of model. For future work it would be interesting to see what a more rigorous investigation could achieve, either by investigating each step closer, looking at the data in a different way, or just plainly actually trying to mimic the practical ABCDE-method as close as possible. One thing we can say for sure is that it is a very complex problem where we need to look at many different indicators, very precisely.

[36]e.g. by tweaking the algorithms or transforming the parameters by the square root or logarithm, etc, different results were acquired.
[37]In fact, quite many papers focus on one specific type of feature.

References

[1] Clťaudio R. Jung Alessandro Parolin, Eduardo Herzer. Semi-automated diagnosis of melanoma through the analysis of dermatological images. 2010. 23rd SIBGRAPI - Conference on Graphics, Patterns and Images.

[2] Mongi Abidi Andreas Koschan. A comparison of median filter techniques for noise removal in color images. 2001. Proc. 7th German Workshop on Color Image Processing.

[3] Sabine Süsstrunk David Alleysson. Spatio-chromatic pca of a mosaiced color image. IS&T Second European Conference on Color in Graphics, Image, and Vision.

[4] James W. Demmel. *Applied Numerical Linear Algebra*. 1997.

[5] Truog Nguyen Gilbert Strang. *Wavelet and Filter Banks*. 1996.

[6] Eliza Hashemi. Mirocalcification detection in mammography using wavelet transform and statistical parameters, 2012.

[7] Barbara Burke Hubbard. *The World According to Wavelets*. 1996.

[8] Dr. Stephen Rice James Church, Dr. Yixin Chen. A spatial median filter for noise removal in digital images.

[9] Martin Lindberg Joran Bergh, Fredrik Ekstedt. *Wavelets*. 2000.

[10] Abdolhossein Sarafzadeh Elham Ghasemi Nima Fassihi, Jamshid Shanbehzadeh. Melanoma diagnosis by the use of wavelet analysis based on morphological operators. 2011. Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol I.

[11] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive wavelet rendering. Technical report, Columbia University, 2009. SIGGRAPH Asia 2009.

[12] Jean-Philippe Thiran Philippe Schmid-Saugeon, Joel Guillod. Towards a computer-aided diagnosis system for pigmented skin lesions. 2002. Computerized Medical Imaging and Graphics 27 (2003) pp.65-78.

[13] Steven L. Eddins Rafael C. Gonzalez, Richard E. Woods. Morphological reconstruction. 2010. Digital Image Processing Using MATLAB.

[14] A. Govardhan Kashyap D Dhruve Ruksar Fatima, Mohammed Zafar Ali Khan. Computer aided multi-parameter extraction system to aid early detection of skin cancer melanoma. 2012. IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.10, October 2012.

[15] M. Stella Atkins Tim K. Lee, David I. McLean. Irregularity index: A new border irregularity measure for cutaneous melanocytic lesions. 2002. Medical Image Analysis 7 (2003) pp.47-64.