CHALMERS


# Sparse representation and image classification with the shearlet transform

## Master's thesis in Engineering Mathematics and Computational science

Robin Andersson

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2017

Master's thesis 2017

# Sparse representation and image classification with the shearlet transform

Robin Andersson


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences

Chalmers University of Technology

Gothenburg, Sweden 2017

Sparse representation and image classification
using the shearlet transform

Robin Andersson



Supervisor: Adam Andersson, Syntronic Software Innovations
Supervisor & Examiner: Mohammad Asadzadeh, Department of Mathematical Sciences

Master's Thesis 2017:
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

**Abstract**

Classical 2D-wavelet transforms have suboptimal compression performance due to its inability to generate sparse representation of discontinuities along lines. This thesis contains investigations of the shearlet transform which in contrast to classical 2D-wavelet transforms is directional. The shearlet transform has optimal compression performance of so called "cartoon-like images" and performs better than wavelet on real images too. Besides image compression the thesis concerns image classification using the shearlet transform as a component of the feature extraction procedure. Images are transformed to symmetric and positive definite (SPD) matrices. The space of SPD matrices is not a linear space but is on the other hand a Riemannian manifold with the structure that provides. For the classification task, a kernel support vector classifier is used that uses the log-Euclidean metric on the space of SPD matrices. The thesis was written at Syntronic Software Innovations.
**Keywords:** Shearlet, wavelet, anisotropy, support vector machine, data classification.

**Acknowledgements**

First I want to thank Adam Andersson who was my supervisor at Syntronic. I will always be grateful for your invaluable guidance and support. You were always available when I needed advice or help. I am very grateful for the patience and enthusiasm you put into my project and it has been encouraging me throughout this work.

I want to thank Mohammad Asadzadeh my supervisor at Chalmers for your help with this thesis and time spent listening to my presentations. Your help have broadened my mathematical knowledge, presentation skills and ability to write profesionally. I am grateful for this.

I would also like to thank Syntronic that offered me this opportunity to work with an interesting and challenging project. This has not only developed my technical knowledge but also myself as a person.

A large amount of the computations in this work have been carried out on the server Ozzy at the department of Mathematical sciences, Chalmers University of Technology. I therefore want to sincerely acknowledge Chalmers for this possibility.

Finally I would like to thank my family and friends for all of your support throughout my studies. You mean the world to me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Pattern recognition has become a major branch of data analysis and considers identification and recognition of patterns in data. Patterns can be prominent but still difficult to identify and therefore methods to enhance these structures are desired. One way to enhance patterns in data is to consider the way data is presented using different types of data transformations. This could for example be the Fourier transform, which reveals present frequencies in a signal, and could be considered as an underlying pattern or structure.

Another common tool for data representation and for signal processing in general, is the framework of *wavelets*. The theory of wavelets was developed throughout the 20th century independently by several mathematicians such as *Alfréd Haar, Ingrid Daubechies* and *Stéphane Mallat* among many others [1]. Wavelets use a time-frequency representation which is different from the Fourier transform which is only localized in frequency. Due to their mathematical richness wavelets have been extensively used to both analyze and compress data since its development.

When considering two-dimensional signals, such as images, we can distinguish between anisotropic and isotropic features. An anisotropic feauture is a property that is directionally dependent. An isotropic feauture however is uniform along all directions. Isotropic and anistropic features are mathematically realized as point singularities and singularities along a curve respectively. Wavelets deal efficiently with isotropic features but fail to efficiently represent anisotropic features [2]. In 2005, attempts to derive new efficient representations lead to several additional frameworks such as the *curvelet*, *contourlet* and *shearlet*. Among these, the shearlet framework that was developed by K. Guo, G. Kutyniok and D. Labate [3] excelled the most due to its ability of providing with optimally sparse representations of anisotropic objects.

Since shearlets were introduced there have been numerous applications of the shearlet to compare its performance against similar frameworks. For example there are denoising applications [4], image compression [5] and feature extraction [6, 7].

The goal of this thesis is to compare the shearlet with wavelets using image classification. We investigate the decay property of the shearlet and wavelet coefficients for different types of images, with and without anisotropic features, to quantify the sparse

representation properties of the transforms. We also demonstrate how the shearlet transform can be used for filtering of images. The main comparison of shearlets and wavelets of this thesis concerns image classification of the MNIST dataset of handwritten digits [8]. It is a standard dataset in machine learning and considered easy. A multitude of classifiers perform better than 99% accuracy on it. The comparison we carry out is between feature extraction with shearlet transform and the Gabor wavelet. The latter is another directional transform, popular by engineers, without the sound theoretical foundations of the shearlet transform. To learn the methods, standard packages were not used, neither for shearlets nor for machine learning. The implementation of shearlets is by nature fast but the training time for the machine learning algorithm very slow. Therefore, 300, 500 and at maximum 700 training data samples are used for each class instead of 6000, and classification performance is far from state of the art for MNIST. In retrospect, training with, e.g., the R-package kernlab would allow training with the full 6000 training dataset and a fair comparison with other methods would be possible. On the other hand, the results show that given the limited amount of training data 700 per digit, the shearlet transform outperforms the Gabor wavelet. Moreover, a large amounts of algorithms have been tested using MNIST which enables comparison of algorithm performance with a multitude of other algorithms. To compare shearlets with wavelets we consider two models where one is based on shearlets and the other one on the *Gabor wavelet*. The Gabor wavelet is a natural choice due to its similar areas of application and the result of its transform resembles similar features. Among engineers the Gabor wavelet is a common tool due to its directionally dependent transform. Both shearlets and Gabor wavelets have succesfully been used for both edge detection [9, 10] and feature extraction [7, 11].

In our model we use the shearlet and Gabor wavelet to construct correlation matrices based upon their transforms of a respective image. This gives each image in the dataset its corresponding correlation matrix. An important property of correlation matrices is that a correlation matrix is a *symmetric positive definite matrix*. The space of symmetric positive definite matrices of size $n \times n$ constitutes a *Riemannian manifold* [12] and this fact means in particular that there is a metric structure and thus a way to measure distance on that manifold. There are several applicable metrics such as the *affine-invariant*, *stein metric* and the *log-Euclidean metric* [13]. We use the log-Euclidean metric due to a reduced computational cost and other theoretical benefits [14] that are outside the scope of this thesis.

The correlation matrices are classified using a classification model known as the *support vector machine* (SVM). There are a numerous different methods available for data classification, some other popular methods are *CART*, *k-means* and *Naive Bayes* [15]. A SVM is a binary classifier based upon finding a *hyperplane* that optimally separates two groups of data. Combining the correlation matrices with the support vector machines, we use the fact that the matrices are elements on a Riemannian manifold, and therefore measure distance between each correlation matrix. To measure distance we need a suitable metric and we combine the log-Euclidean metric with a *Gaussian RBF kernel*. A log-Euclidean based Gaussian RBF kernel have proved to greatly improve classification

results compared to Gaussian RBF kernels based on other metrics in previous work [16].

The outline of this thesis is the following. In Section 2.2 we briefly rehearse some of the major results from wavelet theory. From this section we proceed with the theory regarding shearlets in Section 2.3. In Section 4 we look at support vector machines and how we combine these with positive definite matrices to classify data. Finally in Section 6 we present the results with conclusions in Section 7.

# Chapter 2

# Wavelet and shearlet transforms

This chapter contains the mathematical preliminaries for the linear transforms used in the later chapters. It begins at the very definition of the Fourier transform to proceed with some important concepts from the wavelet theory in Section 2.2. We continue with the extension to wavelets in two dimensions in 2.2.1. The shearlet theory is presented in Section 2.3 followed by a practical implementation in Section 2.3.2. The basis of shearlet theory originates from wavelet theory, which itself is a natural extension from the Fourier analysis. We therefore begin by giving the reader a broad but concise theoretical understanding to these mathematical concepts before introducing the shearlet framework.

## 2.1 Notation and preliminaries

We begin by some introductory notations. A function $f : \mathbb{R}^k \to \mathbb{R}$, $k \geq 1$, is said to belong to $L^p(\mathbb{R}^k; \mathbb{R}) = L^p(\mathbb{R}^k)$, $1 \leq p < \infty$, if it satisfies

$$\int_{\mathbb{R}^k} |f(x)|^p \, \mathrm{d}x < \infty \, .$$

The $L^p(\mathbb{R}^k)$ norm of a function $f$ on $\mathbb{R}^k$ is denoted by

$$||f(x)||_{L^p(\mathbb{R}^k)} \coloneqq \left( \int_{\mathbb{R}^k} |f(x)|^p \, \mathrm{d}x \right)^{\frac{1}{p}} , \quad 1 \leq p < \infty \, .$$

In this thesis we mainly consider functions in $L^2(\mathbb{R}^k)$. We define the inner product on $L^2(\mathbb{R}^k)$ for all $f, g \in L^2(\mathbb{R}^k)$ as

$$\langle f, g \rangle_{L^2(\mathbb{R}^k)} \coloneqq \int_{\mathbb{R}^k} f(x) g(x) \, \mathrm{d}x \, .$$

A space $U$ is said to be complete if the limit of every Cauchy sequence is also in $U$. If we equip a space $U$ with a norm $\| \cdot \|$ such that $U$ is complete with respect to $\| \cdot \|$ then

$U$ is said to be a Banach space. A Banach space is a complete normed vector space. Moreover, if a Banach space $\mathcal{H}$ is equipped with an inner product then $\mathcal{H}$ is a Hilbert space. The space $L^2(\mathbb{R})$ is a Hilbert space, i.e. a complete inner product space. We remind the reader that if $\{u_k\}_{k=0}^n$ is a complete orthonormal basis for some Hilbert space $\mathcal{H}$, then $\{u_k\}_{k=0}^n$ is referred to as a Hilbert basis.

The Fourier transform is used throughout this thesis and plays a central role when we define the shearlet transform.

**Definition 2.1.1.** The continuous Fourier transform of a function $f \in L^p(\mathbb{R}^k)$ is defined as the integral

$$\hat{f}(\xi) \coloneqq \lim_{R\to\infty} \int_{|x|\leq R} f(x) e^{-2\pi i x\cdot\xi} \,\mathrm{d}x\,, \qquad \xi \in \mathbb{R}^k\,. \tag{2.1}$$

All Fourier transforms are denoted by a hat $\hat{f}$ unless otherwise explicitly stated. The definition for the discrete Fourier transform is completely analogous, and is stated next.

**Definition 2.1.2.** Let $\boldsymbol{\xi} = (\xi_1, ..., \xi_m)^T$. The $m$-dimensional discrete Fourier transform of $f(\boldsymbol{n})$, $\boldsymbol{n} = (n_1, ..., n_m)$ is defined as

$$\hat{f}(\boldsymbol{\xi}) \coloneqq \sum_{n_1=0}^{N_1-1} \cdots \sum_{n_m=0}^{N_m-1} f(n_1, n_2, ..., n_m) e^{-i\frac{2\pi}{N_1}n_1\xi_1 - i\frac{2\pi}{N_2}n_2\xi_2 ... - i\frac{2\pi}{N_m}n_m\xi_m}\,.$$

In this thesis we frequently use the *Plancherel theorem*.

**Theorem 1** (Plancherel theorem)**.** *For $f, g \in L^2(\mathbb{R}^k)$, it holds that*

$$\langle f, g\rangle_{L^2(\mathbb{R}^k)} = \langle \hat{f}, \hat{g}\rangle_{L^2(\mathbb{R}^k)}\,.$$

When working with wavelets and shearlets, we often work with Riesz bases.

**Definition 2.1.3.** Let $(V, \|\cdot\|)$ be a Banach space and $\{u_n\}_{n\in\mathbb{N}}$ be a basis of $V$. If there exists positive numbers $A$ and $B$, $A \leq B$, such that for all $\psi = \sum_k c_k u_k$ the following inequalities hold:

$$A\|\psi\|^2 \leq \sum_k |c_k|^2 \leq B\|\psi\|^2\,.$$

Then $\{u_n\}_{n\in\mathbb{N}}$ is called a Riesz basis.

Now when we have defined a Riesz basis it is natural to continue with the concept of frames. Frames play a crucial role in the theory of wavelets and shearlets.

**Definition 2.1.4.** Let $(\mathcal{H}, \|\cdot\|, \langle\cdot,\cdot\rangle)$ be a Hilbert space. A set $\{u_n\}_{n\in\mathbb{Z}} \subset \mathcal{H}$ constitutes a frame if there exist positive constants $A$ and $B$ such that for all $\psi \in \mathcal{H}$ it holds

$$A||\psi||^2 \leq \sum_{n\in\mathbb{Z}} |\langle\psi, u_n\rangle|^2 \leq B||\psi||^2\,.$$

If $A = B = 1$ we have a Parseval frame, this can be put in other words as in the following definition.

**Definition 2.1.5.** Let $(\mathcal{H}, \|\cdot\|, \langle\cdot,\cdot\rangle)$ be a Hilbert space. A set $\{u_n\}_{n\in\mathbb{Z}} \subset \mathcal{H}$ constitutes a Parseval frame if

$$||\psi||^2 = \sum_{n\in\mathbb{Z}} |\langle\psi, u_n\rangle|^2 , \qquad \forall\psi \in \mathcal{H} .$$

We conclude this preliminary chapter by recalling the concept of compact support. If $\Omega$ is a compact set, then a function $f$ has compact support on $\Omega$ if $f$ is zero on the complement of $\Omega$.

These introduced concepts are widely used in the coming chapters. We proceed with the theory regarding wavelets.

## 2.2 Wavelet analysis

The wavelet framework is a natural extension of the Fourier framework. Wavelets have a wide range of applications, especially in signal processing and are useful for data representation and data compression. A wavelet is a local oscillation and can be realized as a short wave with compact support. Wavelets differ mainly from the Fourier transform due to its localization in both frequency and time, whilst the Fourier transform is only localized in frequency. Even though the wavelets are localized in frequency and time, one cannot simultaneously tell the exact frequency response at a certain point in time. This limitation is related to a very famous postulate in quantum mechanics and analysis known as Heisenberg's uncertainty principle.

To put things in context we begin with a formal definition of a wavelet system.

**Definition 2.2.1.** A discrete wavelet system in $L^2(\mathbb{R})$ with mother wavelet $\psi \in L^2$, is the family of functions given by

$$\mathcal{W}(\psi) \coloneqq \left\{\psi_{j,k} = 2^{\frac{j}{2}}\psi(2^j \cdot -k) \,|\, j,k \in \mathbb{Z}\right\} . \tag{2.2}$$

Ideally one desires $\mathcal{W}(\psi)$ to constitute an ON-basis of $L^2(\mathbb{R})$ but this is not a requirement for a discrete wavelet system, and not possible for most interesting discrete wavelet families. If $\psi \in L^2(\mathbb{R})$ makes $\mathcal{W}(\psi)$ an ON-basis, then $\psi$ is called an orthonormal wavelet. Thus if $\psi \in L^2(\mathbb{R})$ is an orthonormal wavelet and $\mathcal{W}(\psi)$ is the family of functions defined by (2.2), then any function $f \in L^2(\mathbb{R})$ can be expanded as

$$f = \sum_{j,k\in\mathbb{Z}} w_{j,k}\psi_{j,k} . \tag{2.3}$$

Here $\{w_{j,k}\}_{j,k\in\mathbb{Z}} \in \mathbb{R}$ are known as the wavelet coefficients. If $\mathcal{W}(\psi)$ does not constitute an ON-basis, but instead if it constitutes a frame, the coefficients $w_{j,k}$ are replaced by the inner products $\langle f, \psi_{j,k}\rangle$.

The wavelet coefficients are of great importance while analyzing a signal. When transforming a signal using the wavelet transform, a set of coefficients is obtained and these contain information about the given signal.

**Theorem 2.** *If $f(t) \in L^2(\mathbb{R})$, then the wavelet coefficient $w_{j,k}$ is given by*

$$w_{j,k} = \int_{\mathbb{R}} f(t)\psi_{j,k}(t)\,\mathrm{d}t\,. \tag{2.4}$$

A wavelet coefficient $w_{j,k}$ corresponds to the norm of a projection of the signal onto $\psi_{j,k}$, thus we obtain one coefficient for each pair $(j,k)$. The expression for calculating the wavelet coefficients in the continuous case is completely analogous to the continuous Fourier transform.

Note that if $\psi_{j,k}$ is replaced by $e^{-2\pi i\omega t}$ in (2.4) this is exactly the Fourier transform. Yet we have not given an explicit expression for any wavelet $\psi$. Compared to Fourier analysis, the basis functions $\psi_{j,\,k}$ do not always have explicit expressions. There are also several restrictions on $\psi$ to be able to classify $\psi$ as a wavelet. When constructing a wavelet system one usually picks $\psi$ depending on the features of the signal being analyzed. The reasoning behind this is that the wavelet coefficients are basically a convolution between the signal and the wavelet (see Equation (2.4)). Therefore, different wavelets may correlate differently with $x$ giving different results. We proceed by showing how one can construct $\psi$.

If $\psi \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ one can formulate the following conditions

$$\int_{\mathbb{R}} \psi(t)\,\mathrm{d}t = 0\,, \qquad \int_{\mathbb{R}} |\psi(t)|^2\,\mathrm{d}t = 1\,,$$

which is equivalent to $\psi$ having zero mean and $L^2$-norm equal to one.

The question now is how one can construct a function $\psi$ that satisfies all of the desired properties such that the transformation can be realized practically. To date, the general machinery to construct a discrete wavelet transform is using *multiresolution analysis* (MRA). This is a method developed in the late 1980s by Stephene Mallat and Yves Meyer. MRA also forms one of the foundations to the *fast wavelet transform*.

**Definition 2.2.2.** A multiresolution analysis of $L^2(\mathbb{R})$ is a sequence of closed subspaces that satisfies the following properties:

1. $V_j \subset V_{j+1}$ for all integers $j$,
2. $f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1}$ for all integers $j$.
3. $f(t+1) \in V_j \Leftrightarrow f(t) \in V_j$ for all integers $j$.
4. The set $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$ and $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$.
5. There exists a function $\phi \in V_0$ such that its integer shifts forms a Riesz basis for $V_0$, i.e. that $\{\phi(t-k)\}_{k\in\mathbb{Z}}$ is a Riesz basis for $V_0$.

A MRA leads to filter banks and we now explain this. Since the integer shifts of $\phi$ forms a Riesz basis for $V_0$, we call $\phi$ scaling function and say that $\phi$ generates

$V_0$. Similarly if we define $\phi_{j,k} = 2^{\frac{j}{2}}\phi(2^j t - k)$ then the integer shifts of $\phi_{j,k}$ span $V_j$. Moreover, since $V_0 \subset V_1$ there is a sequence $\{g_n\}_n \subset \mathbb{R}$ that satisfies

$$\phi(t) = \sqrt{2}\sum_{n=0}^{N-1} g_n \phi(2t-n)\,. \tag{2.5}$$

Note that $\phi(2t-n) \in V_1$. That $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$ means that for any $f \in L^2(\mathbb{R})$ there exists a sequence $f_k \in V_k$ such that $\|f - f_k\| \to 0$ as $k \to \infty$. Let $f_j$ and $f_{j+1}$ be approximations of $f$ at scale $j$ and $j+1$ respectively, then $d_j = f_{j+1} - f_j \in V_{j+1}$ since $f_j \in V_j \subset V_{j+1}$. We introduce the space $W_j$ of functions $d_j$ as the details lost of a function $f$ between the approximations at scale $j$ and $j+1$, that is, $W_j = V_{j+1} \ominus V_j$, for $\ominus$ denoting the orthogonal difference. The space $W_0$ is the orthogonal complement to $V_0$ and $W_0 \subset V_1$. For $\oplus$ denoting the orthogonal sum we have $V_1 = W_0 \oplus V_0$.

We call $\psi$ a wavelet if $W_0 \subset V_1$ is the space generated by the integer shifts of the mother wavelet, namely, the space spanned by $\{\psi(t-k)\}_{k\in\mathbb{Z}}$. Moreover $V_1 = W_0 \oplus V_0$ and each $f_1 \in V_1$ can uniquely be written as $f_1 = f_0 + d_0$ where $f_0 \in V_0$ and $d_0 \in W_0$.

With the recently stated results, we define the mother wavelet $\psi$ using $\phi$ as

$$\psi(t) \coloneqq \sqrt{2}\sum_{n=0}^{N-1}(-1)^n g_{N-1-n}\phi(2t-n) = \sqrt{2}\sum_{n=0}^{N-1} h_n \phi(2t-n)\,, \tag{2.6}$$

Equation (2.5) and (2.6) are known as the refinement equations. The sequences $\{g_n\}_{n=0}^{N-1} \subset \mathbb{R}$ and $\{h_n\}_{n=0}^{N-1} \subset \mathbb{R}$ are called scaling sequence and wavelet sequence respectively. The scaling function $\phi$ requires to have integral one due to specific approximation properties [17]. Moreover from (2.5) it follows that

$$1 = \int_{\mathbb{R}} \phi(t)\,\mathrm{d}t = \int_{\mathbb{R}} \sqrt{2}\sum_{n=0}^{N-1} g_n \phi(2t-n)\,\mathrm{d}t = \sqrt{2}\sum_{n=0}^{N-1} g_n \int_{\mathbb{R}} \phi(2t-n)\,\mathrm{d}t\,.$$

Note that

$$\int_{\mathbb{R}} \phi(2t-n)\,\mathrm{d}t = \int_{\mathbb{R}} \phi(s)\frac{1}{2}\,\mathrm{d}s = \frac{1}{2}\,,$$

since the scaling function integrates to 1 on $\mathbb{R}$. This gives us

$$1 = \sqrt{2}\sum_{n=0}^{N-1} g_n \int_{\mathbb{R}} \phi(2t-n)\,\mathrm{d}t = \frac{1}{\sqrt{2}}\sum_{n=0}^{N-1} g_n \Rightarrow \sum_{n=0}^{N-1} g_n = \sqrt{2}\,.$$

In a similar way we obtain another constraint on $h_n$ since

$$0 = \int_{\mathbb{R}} \psi(t)\,\mathrm{d}t = \int_{\mathbb{R}} \sqrt{2}\sum_{n=0}^{N-1} h_n \phi(2t-n)\,\mathrm{d}t \Rightarrow \sum_{n=0}^{N-1} h_n = 0\,.$$

The coefficients $g_n$ are calculated multiplying (2.5) by $\phi(2t-n)$ and using the orthogonality conditions to obtain

$$\sqrt{2}\int_{\mathbb{R}} \phi(t)\phi(2t-n)\,\mathrm{d}t = g_n\,. \tag{2.7}$$

When $g_n$ is known we also know $h_n$ since $h_n = (-1)^n g_{N-1-n}$.

Before we proceed with the theory regarding filter banks we illustrate the theory above with the most basic wavelet.

**Example 1** (The Haar wavelet)**.** The Haar wavelet was already developed in the early 1900s by Alfréd Haar long before any general ideas of wavelet theory has been established. This wavelet is the most simple one and we proceed by deriving the coefficients $g_n$, $h_n$ and the wavelet $\psi$ starting from the scaling function $\phi$. The scaling function is defined as

$$\phi(t) := \begin{cases} 1\,, & 0 \leq t \leq 1\,, \\ 0\,, & \text{else}\,. \end{cases}$$

To derive $\psi$ we need $g_n$. Using Equation (2.7) we obtain $g_0 = g_1 = \frac{1}{\sqrt{2}}$ and the rest of the coefficients are equal to zero. This gives us $h_0 = \frac{1}{\sqrt{2}}$ as well as $h_1 = -h_0 = -\frac{1}{\sqrt{2}}$. Combining these results with (2.6) we obtain the Haar mother wavelet

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

The Haar basis is shown in Figure 2.1 To be able to express a signal $f$ in the wavelet basis one requires the scaling and wavelet coefficients. The calculation is a recursive process where through the orthogonality relations and the refinement equations one can see the equivalence between these calculations and passing the function through a filter bank. That is, the calculation of coefficients is equivalent to passing a signal $f$ through a series of filters.

Now that we have established several underlying keystones from wavelet analysis we are ready to see the practical calculations to obtain the wavelet coefficients. A filter bank is constructed as a cascade of low-pass and high-pass filters. Assume we have a signal $x$ of $N$ samples, for simplicity also assume $N = 2^M$ for some $M \in \mathbb{N}$. The signal $x$ is then passed through the low-pass filter and high-pass filter respectively. If we denote any of these filters by $f$ then mathematically this is realized as the convolution

$$y_f(n) = x * f(2n) = \sum_{k=1}^{N} x(k) f(2n-k)\,.$$

After each filter the filtered signal is downsampled which means that each element with odd index is removed, this downsampling is denoted by the operator $\downarrow 2$. The process of filtering and downsampling can be denoted as

$$y = \downarrow 2(x * f) = (y_f(0), y_f(2), ..., y_f(N))\,.$$

The coefficients $y_f$ are exactly the wavelet coefficients if $f$ is the downsampling high-pass filter $\{h_n\}_{n=1}^N$. From the low-pass filter $\{g_n\}_{n=1}^N$ we obtain similarly the scaling

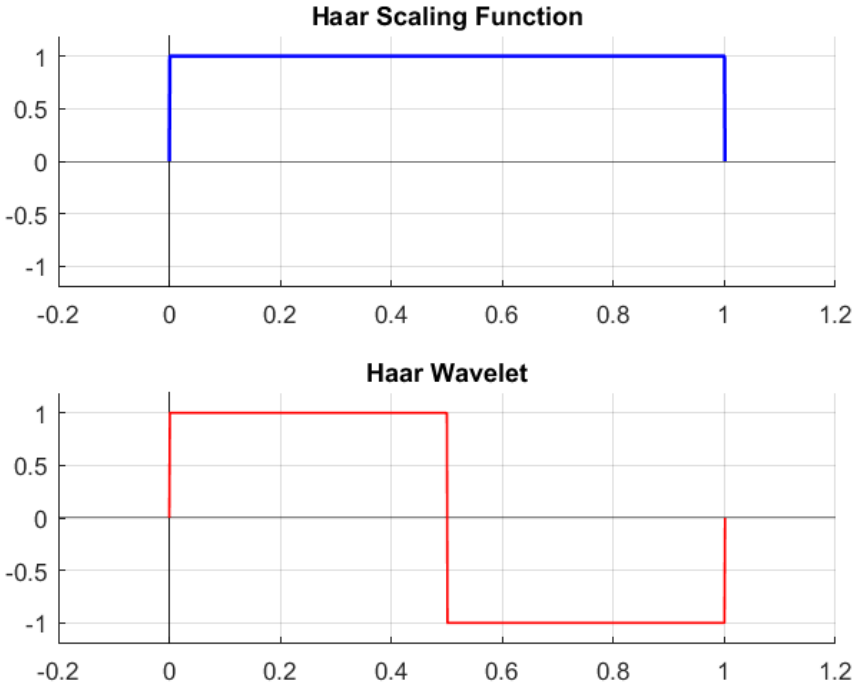


**Figure 2.1:** This figure illustrates the scaling function $\phi$ and wavelet $\psi$ for the Haar basis.

coefficients. Due to $\downarrow 2$ one has $N/2$ coefficients from each filter. The wavelet coefficients are stored while the remaining scaling coefficients - which are equivalent to the filtered signal - are then passed through another set of filters (one high-pass and one low-pass filter respectively). This results in another set of $N/4$ wavelet- and scaling coefficients. This process is repeated. We notice that this halves the number of samples for each application of $\downarrow 2$. Hence if $N = 2^n$ for some positive integer $n$ we can downsample exactly $\log(N)/\log(2)$ times until we have only one remaining sample. This results in exactly $N$ coefficients since we obtain $N/2 + N/4 + ... + N/2^{n-1} + 2 = N$ wavelet coefficients in total. The entire process can be realized graphically, see Figure 2.2.

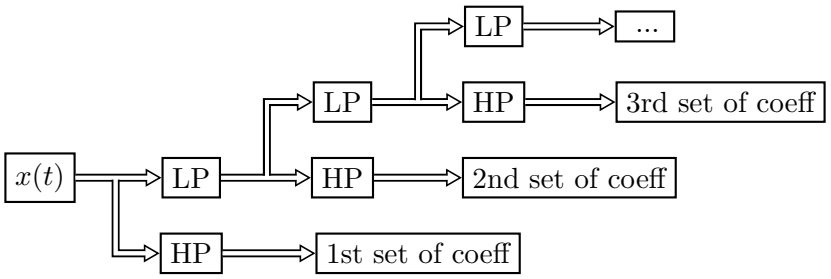


**Figure 2.2:** This figure illustrates how the signal is passed through a series of filters. LP and HP denotes low-pass and high-pass filter respectively. After each high-pass filter we extract a set of wavelet coefficients, this procedure proceeds for a finite amount of filters depending on the length of the signal. Note that the signal after each filter is also downsampled but not illustrated in the figure.

**Example 2.** To illustrate with an example we take the Haar Wavelet. As derived earlier this choice of basis results in filter coefficients $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ and $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$ for the low-pass and high-pass filter respectively. Assume for simplicity that we have the signal $x = (-9, -4, 1, 0, 1, 4, 9, 16)$. To calculate the coordinates of $x$ in the Haar basis we have to "pass $x$ through" each filter and repeat this process until we have obtained all the wavelet coefficients. The coefficients from each filter at each stage is then

$$x \to \begin{cases} \text{LP}: \ \frac{1}{\sqrt{2}}(-13, 1, 5, 25) & \to \frac{1}{2}(-12, 30) & \to \frac{1}{\sqrt{2}}(9) \\ \text{HP}: \ \frac{1}{\sqrt{2}}(-5, 1, -3, -7) & \to \frac{1}{2}(-7, -10) & \to \frac{1}{\sqrt{2}}(-21) \end{cases}.$$

This means that we have the wavelet coefficients

$$w = \left(-\frac{5}{\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{3}{\sqrt{2}}, -\frac{7}{\sqrt{2}}, -\frac{14}{2}, -\frac{20}{2}, -\frac{21}{\sqrt{2}}, \frac{9}{\sqrt{2}}\right).$$

The calculation can also be realized as the matrix vector product:

$$w = \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{2} & -\sqrt{2} \end{pmatrix} \begin{pmatrix} -9 \\ -4 \\ 1 \\ 0 \\ 1 \\ 4 \\ 9 \\ 16 \end{pmatrix} = \begin{pmatrix} \frac{9}{\sqrt{2}} \\ -\frac{21}{\sqrt{2}} \\ -7 \\ -10 \\ -\frac{5}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ -\frac{3}{\sqrt{2}} \\ -\frac{7}{\sqrt{2}} \end{pmatrix}.$$

The coefficients $w$ contain information about the frequencies characterizing the signal. However, recall that the obtained results depend strongly on the choice of wavelet

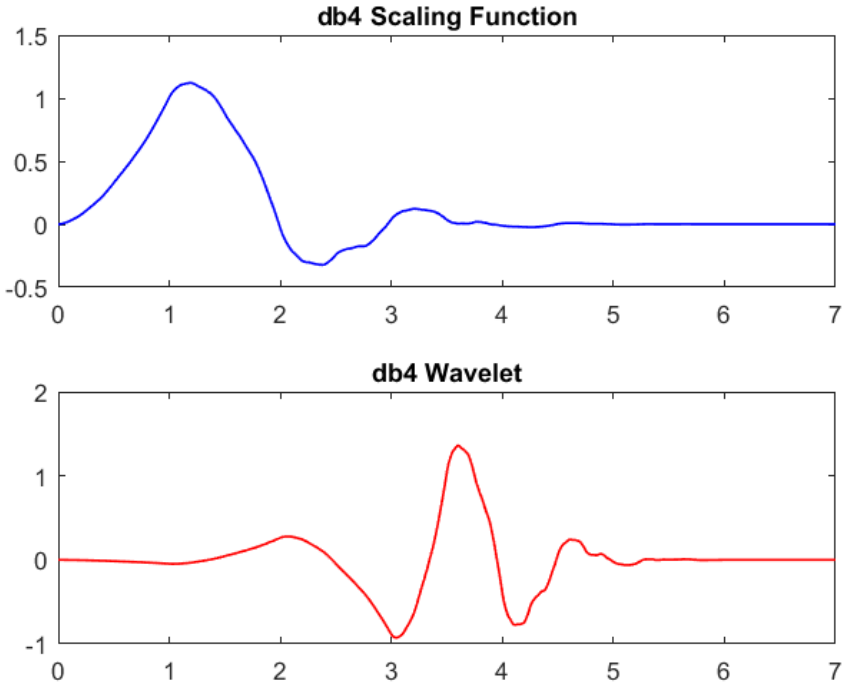


**Figure 2.3:** This figure illustrates the scaling function $\phi$ and wavelet $\psi$ for the db4 basis.

since the wavelet coefficients can be visualized as a projection of the signal onto the wavelet itself. The Haar basis is the simplest one and the coefficients that comes along with it can be visualized as repeated averages and differences times a factor. Its simplicity does not necessary imply that it is inadequate, however, it is not the most practically applied wavelet.

**Example 3** (Daubechies wavelet)**.** Another common family of wavelets is *the Daubechies wavelets* based on the work of Ingrid Daubechies. These wavelets are favoured mainly due to their properties related to *vanishing moments*. The concept of *vanishing moments* is a term related to compression functionality. The Daubechie wavelets can be chosen with a different amount of vanishing moments, giving different filter coefficients and different compression qualities. The reader who wants to learn more about this is recommended to consult [17]. The Daubechie scaling- and wavelet functions do not have explicit expression thus one has to pursue more advanced methods to obtain the filter coefficients. For simplicity we illustrate the transformation of a signal $x = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ with the Daubechie wavelet called *D4* (or db4). The function D4 is illustrated in Figure 2.3. This basis has low-pass coefficients [18]

$$h = (h_0, h_1, h_2, h_3) = \frac{1}{4\sqrt{2}}(1+\sqrt{3}, 3+\sqrt{3}, 3-\sqrt{3}, 1-\sqrt{3}),$$

and high-pass coefficients

$$g = (g_0, g_1, g_2, g_3) = (h_3, -h_2, h_1, -h_0)\,.$$

The transformation matrix for this signal of 8 samples is

$$D = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \end{pmatrix}$$

Theoretically signals are assumed to be of infinite length, however the filter-banks assume signals of finite length. One way to deal with this is to extend the signal by modifying its length. For example a periodic extension assumes that the signal repeats itself. There are many other extension techniques. One popular method is to assume a periodic extension. That is if $x$ is of size 8 and we use the filter given by $D$, one would add elements from $x$, starting from its first value $x_0$, such that it matches the filter size. For example we extend $x$ by using the first two elements to have a length equal to 10, thus $\widetilde{x} = (x_0, x_1, x_2, ..., x_6, x_7, x_0, x_1) = (x, x_0, x_1)$.

### 2.2.1 Two-dimensional wavelets

So far we have only covered applications of wavelets for one-dimensional signals. However two-dimensional signals (or higher), such as images, are well suited for wavelet transformations. Before we leave the wavelet theory and proceed with shearlets we briefly introduce the idea behind two-dimensional wavelets.

Previously when we introduced the MRA, we encountered the spaces $V_j$ and $W_j$. We wil briefly show this result can be extended for a two-dimensional scenario. Let $\otimes$ denote a tensor product, if $\{V_j\}$ is a MRA of $L^2(\mathbb{R})$ let $V_j^2 = V_j \otimes V_j$. Then $\{V_j^2\}_j$ is a MRA of $L^2(\mathbb{R}^2)$.

This holds true because if $H = H_1 \otimes H_2$ and $H, H_1, H_2$ are Hilbert spaces and $\{u_j\}_j$, $\{v_j\}_j$ are basis of $H_1$ and $H_2$, respectively. Then $\{u_i \otimes v_j\}_{i,j}$ is a basis of $H$. By a similar argument this holds for $W_j^2$. Therefore by using the one-dimensional MRA $V_j$ we obtain a two-dimensional variant by construcitng $V_j^2$.

To define two-dimensional wavelets, we let $\phi$ and $\psi$ denote the one dimensional scaling and wavelet function from the MRA $\{V_j\}$. Using the one-dimensional functions

we define

$$\begin{aligned}\Phi(x,y) &= \phi(x)\phi(y)\,,\\ \Psi^H(x,y) &= \phi(x)\psi(y)\,,\\ \Psi^V(x,y) &= \psi(x)\phi(y)\,,\\ \Psi^D(x,y) &= \psi(x)\psi(y)\,,\end{aligned}$$

where the subscripts $H, V$ and $D$ are indications of "direction" in the two-dimensional signals, i.e. horizontal, vertical and diagonal. The function $\Phi$ is the scaling function, while the rest are wavelet functions. If $\kappa = \{H, V, D\}$ we define

$$\Psi^{\kappa}_{j,n,m}(x,y) \coloneqq \frac{1}{2^j}\Psi^{\kappa}\left(\frac{x-2^j n}{2^j}, \frac{y-2^j m}{2^j}\right), \tag{2.8}$$

and similarly for $\Phi_{j,\,n,\,m}$. Then $\{\Psi^H_{j,\,n,\,m}, \Psi^V_{j,\,n,\,m}, \Psi^D_{j,\,n,\,m}\}_{j,\,n,\,m}$ is a frame for $L^2(\mathbb{R}^2)$.

Wavelet coefficients are obtained using the functions $\Psi^{\kappa}$ giving coefficients for each direction in $\kappa$.

### 2.2.2 The Gabor wavelet

The final wavelet that we remark on is the Gabor wavelet. The Gabor wavelet is a non-orthogonal wavelet and is frequently used as a tool for feature extraction due to its capability of detecting edges in images [19, 20]. A Gabor wavelet is the product of a gaussian function and a complex exponential. More formally, we can write a one-dimensional Gabor function $g(x) : \mathbb{R} \to \mathbb{C}$, centered around $x_0 \in \mathbb{R}$, as

$$g_{a,\,\xi}(x) = A\exp\left(-\frac{1}{a^2}(x-x_0)^2\right)\exp\left(-i\xi(x-x_0)\right), \qquad x \in \mathbb{R}\,.$$

where $A$ is a normalization constant such that $g_{a,\,\xi}(x)$ integrates to 1 on $\mathbb{R}$. Here $a, \xi \in \mathbb{R}$ are parameters.

To construct a two-dimensional wavelet we require a two-dimensional function. This is obtained in a similar manner demonstrated in Section 2.2.1. Thus, using the one-dimensional function we obtain

$$g_{a,\,\xi_1,\,\xi_2}(x,y) = g_{a,\,\xi_1}(x)g_{a,\,\xi_2}(y)\,, \qquad x, y \in \mathbb{R},. \tag{2.9}$$

Finally a two-dimensional wavelet is obtained using Equation (2.8) with $g_{a,\,\xi_1,\,\xi_2}(x,y)$ as a generating function. A plot of the basis can be seen in Figure 2.4.

The function given by (2.9) is also referred to as a filter [12]. A representation of an image $I$ using the wavelets is obtained by the convolution $I * g_{a,\,\xi_1,\,\xi_2}$. We can move from rectilinear coordinates $(x, y)$ to polar coordinates $\rho(\cos\theta, \sin\theta)$ where $\rho = \sqrt{x^2+y^2}$ and $\theta$ is the angle of rotation from the $x$-axis. Then we can interpret the wavelet as a directional wavelet where $\theta$ specifies its orientation. By choosing $\xi$ and $\theta$ properly, we can construct a filter bank based on the wavelet such that it covers the entire frequency band. Then from the convolution between an image $I$ and a two-dimensional filter, we

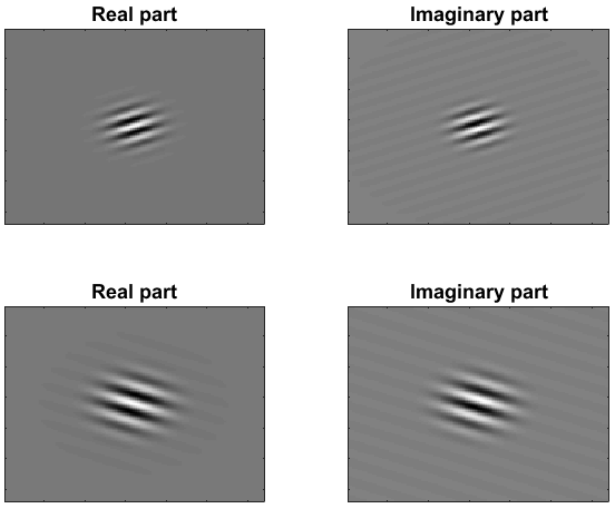


**Figure 2.4:** This figure illustrates the real part and imaginary part of two Gabor wavelets. Note that the imaginary and real part differ in phase.

obtain information of edges in $I$ along a certain direction specified by $\theta$. Edges that are normal to the direction specified by $\theta$ correlate the most, while edges perpendicular to $\theta$ do not not correlate at all.

This wavelet is fundamentally similar to the shearlet that we introduce in the coming section. In Section 4 when we discuss models for classification, the wavelet plays a major role in the implementation of our classification models. This marks the end of the wavelet theory and we proceed with a multiscale framework, namely, shearlets.

## 2.3 Shearlets

The advantage of using shearlets over wavelets is due to its capability of dealing with anisotropic features. Anisotropic features are for example edges in an image and these can be realized mathematically as singularities along a line. Approximations provided by wavelets are not optimally sparse for images containing edges. However optimally sparse approximations in the presence of edges can be obtained using shearlets. The set of *cartoon-like* images, is the set of functions $f$ that are $C^2$ everywhere apart from piecewise $C^2$ edges. Basically $f = f_0 + f_1\chi_B$ is cartoon-like if $f_i \in \mathcal{C}^2$, $i = 0, 1$ with compact support on $B$ and $\partial B$ is a closed $\mathcal{C}^2$-curve. Denote $f_N$ as the shearlet approximation of a *cartoon-like* image $f$ by using the $N$ largest shearlet coefficients, the error between $f$ and $f_N$ satisfies the following decay rate [21]:

$$||f - f_N||^2_{L^2(\mathbb{R}^2)} \leq CN^{-2}(\log N)^3\,, \qquad N \in \mathbb{N}\,,$$

for some $C > 0$. If we do a similar comparison of approximating $f$ by $f_N$ using the best $N-$term approximation of wavelet coefficients, we have the following decay rate

$$\|f - f_N\|^2_{L^2(\mathbb{R}^2)} \leq CN^{-1}\,, \qquad N \in \mathbb{N}\,.$$

When working with wavelet systems one desires to work with orthonormal basis. However in practice it is nearly impossible to create a wavelet system that constitutes such a basis. This holds true also for shearlet systems, however it is possile to create a shearlet system that constitutes a Parseval frame for $L^2(\mathbb{R}^2)$.

### 2.3.1 Shearlet system

We begin by defining a shearlet system without explicitly defining the function generating the system. Denote $\mathbb{R}^+ = [0, \infty)$ as the set of all non-negative real numbers. We begin by defining a continuous shearlet system and proceed with a discrete system.

**Definition 2.3.1.** Consider a function $\psi \in L^2(\mathbb{R}^2)$ and let $\mathbb{S} = (\mathbb{R}^+ \times \mathbb{R}) \times \mathbb{R}^2$. We define a continuous shearlet system, with respect to $\psi$ and $\mathbb{S}$ by

$$\mathcal{SH}(\psi; \mathbb{S}) \coloneqq \left\{ \psi_{a,s,t} = a^{-\frac{3}{4}} \psi(A_a^{-1} S_s^{-1}(x-t)) = a^{-\frac{3}{4}} \psi\left( \begin{pmatrix} \frac{1}{a} & -\frac{s}{a} \\ 0 & \frac{1}{\sqrt{a}} \end{pmatrix} (x-t) \right) \Big| \, (a,s,t) \in \mathbb{S} \right\}. \tag{2.10}$$

Here $A_a$ and $S_s$ denote

$$A_a = \begin{pmatrix} a & 0 \\ 0 & \sqrt{a} \end{pmatrix}, a \in \mathbb{R}^+\,, \qquad S_s = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}, s \in \mathbb{R}\,.$$

The parameters $a$ and $s$ are dilation and shearing parameter respectively. A discrete shearlet system is completely analogous to the system given by (2.10) except that the parameters $(a, s, t)$ range over some discrete lattice $\Lambda \subset \mathbb{S} \subset (\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}^2$. Suitable values of $a, s$ and $t$ are discussed later.

If $\psi \in L^2(\mathbb{R}^2)$ is a shearlet, we define a discrete shearlet system by

$$\mathcal{SH}(\psi; \Lambda) \coloneqq \left\{ \psi_{a,s,t} = a^{-\frac{3}{4}} \psi(A_a^{-1} S_s^{-1}(\cdot - t)) = a^{-\frac{3}{4}} \psi\left( \begin{pmatrix} \frac{1}{a} & -\frac{s}{a} \\ 0 & \frac{1}{\sqrt{a}} \end{pmatrix} (\cdot - t) \right) \Big| \, (a,s,t) \in \Lambda \right\}.$$

So far we have said nothing about what the shearlet $\psi$ actually is, except that $\psi \in L^2(\mathbb{R}^2)$.

**Definition 2.3.2** (Classical shearlet)**.** Let $\psi_1 \in L^2(\mathbb{R})$ be a discrete wavelet in the sense that its Fourier transform satisfies the discrete Calderón condition

$$\sum_{j \in \mathbb{Z}} |\hat{\psi}_1(2^{-j}\omega)|^2 = 1\,, \qquad \text{for a.e. } \omega \in \mathbb{R}\,,$$

and $\hat{\psi}_1 \in \mathcal{C}^\infty(\mathbb{R})$, $\operatorname{supp}(\hat{\psi}_1) \subseteq [-\frac{1}{2}, -\frac{1}{16}] \cup [\frac{1}{16}, \frac{1}{2}]$. We also define $\psi_2 \in L^2(\mathbb{R})$ as a bump function ($\mathcal{C}^\infty$ and compact support) in the sense that it satisfies $\hat{\psi}_2 \in \mathcal{C}^\infty(\mathbb{R})$, $\operatorname{supp}(\hat{\psi}_2) \subseteq [-1, 1]$ and

$$\sum_{k=-1}^{1} |\hat{\psi}_2(\omega + k)|^2 = 1\,, \qquad \text{for a.e. } \omega \in [-1, 1]\,.$$

Then, define $\psi \in L^2(\mathbb{R}^2)$ to be the function defined by its Fourier transform

$$\hat{\psi}(\xi) = \hat{\psi}_1(\omega_1)\hat{\psi}_2\left(\frac{\omega_2}{\omega_1}\right), \qquad \omega = (\omega_1, \omega_2) \in \mathbb{R}^2\,.$$

A function $\psi$ satisfying these conditions is called a classical shearlet.

A discrete shearlet system cannot compose a basis, however it is possible to construct a discrete shearlet system that constitutes a frame by choosing $\Lambda$ appropriately. Such shearlet systems are derived, e.g. by choosing $\Lambda = \{(2^{-j}, -k, S_{-k}A_{2^{-j}}m) : j, k \in \mathbb{Z}, m \in \mathbb{Z}^2\}$, see [21]

**Theorem 3.** *Let $\psi \in L^2(\mathbb{R}^2)$ be a classical shearlet and $\Lambda = \{(2^{-j}, -k, S_{-k}A_{2^{-j}}m) : j, k \in \mathbb{Z}, m \in \mathbb{Z}^2\}$. Then $\mathcal{SH}(\psi; \Lambda)$ is a Parseval frame for $L^2(\mathbb{R}^2)$.*

### 2.3.2 Implementation of the shearlet transform

There are several ways to implement this transformation, here we focus on the *fast finite shearlet transformation* (FFST). The following is only an outline of the work fully described in [22].

We begin by constructing the shearlets as a composition of functions satisfying several properties such that the final product satisfies all the requirements above. We define $v : \mathbb{R} \to \mathbb{R}$ by

$$v(x) := \begin{cases} 0 & x < 0\,, \\ 35x^4 - 84x^5 + 70x^6 - 20x^7 & 0 \le x \le 1\,, \\ 1 & x > 1\,. \end{cases}$$

The function $v$ is symmetric around $(\frac{1}{2}, \frac{1}{2})$, and is the same function used to define the Meyer wavelet. Moreover, we define the function $b : \mathbb{R} \to \mathbb{R}$ by

$$b(\omega) := \begin{cases} \sin(\frac{\pi}{2}v(|\omega| - 1)) & 1 \le |\omega| \le 2\,, \\ \cos(\frac{\pi}{2}v(\frac{1}{2}|\omega| - 1)) & 2 < |\omega| \le 4\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

The function $b$ is positive, real-valued symmetric with compact support on the interval $[-4, -1] \cup [1, 4]$.

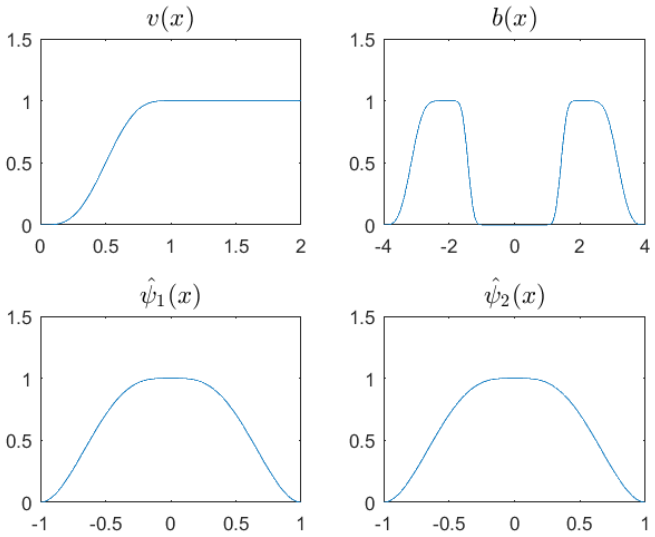


**Figure 2.5:** This figure illustrates the functions $v, b, \hat{\psi}_1$ and $\hat{\psi}_2$.

Using $v$ and $b$ we construct a classical shearlet. Define the function $\psi_1 : \mathbb{R} \to \mathbb{R}$ of Definition 2.3.2 via its Fourier transform, given by

$$\hat{\psi}_1(\omega) := \sqrt{b^2(2\omega) + b^2(\omega)}\,.$$

This function has compact support on $[-4, -\frac{1}{2}] \cup [\frac{1}{2}, 4]$. In a similar way we define the function $\psi_2 : \mathbb{R} \to \mathbb{R}$ of Definition 2.3.2 by

$$\hat{\psi}_2(\omega) := \begin{cases} \sqrt{v(1+\omega)} & \omega \leq 0\,, \\ \sqrt{v(1-\omega)} & \omega > 0\,. \end{cases}$$

As in Definition 2.3.2, $\psi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is given by

$$\hat{\psi} = \hat{\psi}_1(\omega_1)\hat{\psi}_2\left(\frac{\omega_2}{\omega_1}\right).$$

We plot the functions $v, b, \hat{\psi}_1, \hat{\psi}_2$ in Figure 2.5 and $\hat{\psi}$ in Figure ... . We consequently obtain

$$\begin{aligned} \hat{\psi}_{a,s,t}(\omega) &= a^{-\frac{3}{4}}\mathcal{F}\left(\psi\left(\begin{pmatrix} \frac{1}{a} & -\frac{s}{a} \\ 0 & \frac{1}{\sqrt{a}} \end{pmatrix}(\cdot - t)\right)\right)(\omega) \\ &= a^{-\frac{3}{4}} e^{-2\pi i\langle \omega, t\rangle}(a^{-\frac{3}{2}})^{-1}\hat{\psi}\left(\begin{pmatrix} a & 0 \\ s\sqrt{a} & \sqrt{a} \end{pmatrix}\omega\right) \\ &= a^{\frac{3}{4}} e^{-2\pi i\langle \omega, t\rangle}\hat{\psi}_1(a\omega_1)\hat{\psi}_2\left(a^{-\frac{1}{2}}\left(\frac{\omega_2}{\omega_1} + s\right)\right). \end{aligned}$$

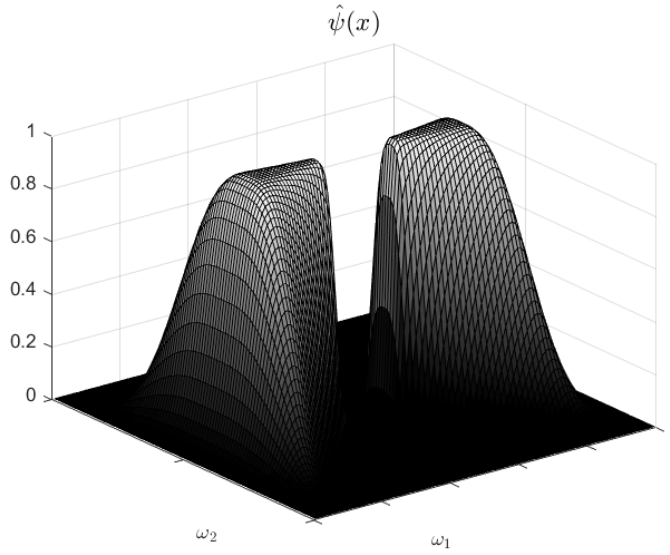


**Figure 2.6:** This figure illustrates the function $\hat{\psi}$.

Using Plancherel's theorem, we denote the shearlet transform of $f$ by $\mathscr{SH}(f)$ which is given by

$$\mathscr{SH}(f)(a,s,t) = \langle f, \psi_{a,s,t}\rangle = \langle \hat{f}, \hat{\psi}_{a,s,t}\rangle = \int_{\mathbb{R}^2} \hat{f}(\omega)\overline{\hat{\psi}_{a,s,t}(\omega)}\,\mathrm{d}\omega$$

$$= a^{\frac{3}{4}}\mathcal{F}^{-1}\left(\hat{f}(\omega)\hat{\psi}_1(a\omega_1)\hat{\psi}_2\left(a^{-\frac{1}{2}}\left(\frac{\omega_2}{\omega_1}+s\right)\right)\right)(t)\,.$$

This forms the basis for the following implementation of the fast finite discrete shearlet transform.

### 2.3.3 Cone-adapted shearlets

With the current construction there is an issue with the shearlet transform for a certain type of functions that are dense along a specific axis. This issue is referred to as a directional bias. The image shown in Figure 2.7 shows the support induced by the current construction of shearlets and for some parameters $(a,s,t)$.

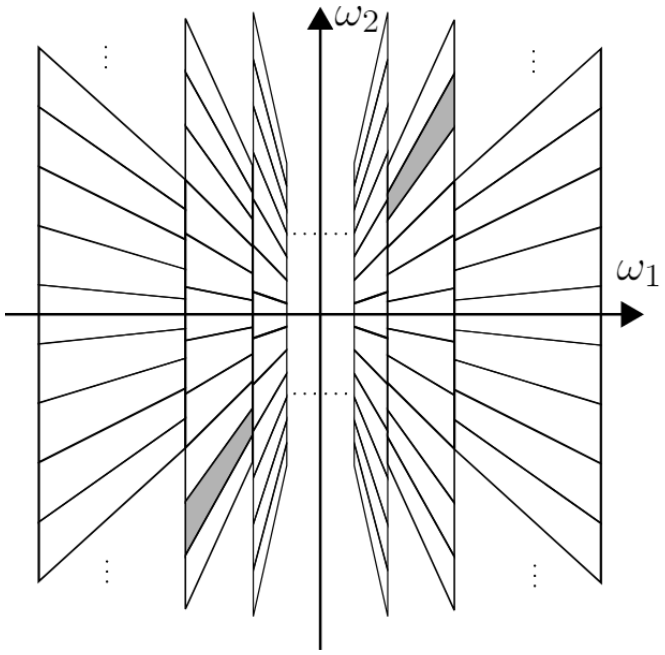


**Figure 2.7:** The tilings induced by the classical shearlets. Note how the support behaves near the vertical axis. Original image by author[1]. Note that of course the supports are not disjoint, the image is simplified to illustrate the support for some values of $(a, s, t)$.

The more concentrated a function is along an axis, the more information of the function is only perceptible in $\mathscr{SH}(f)(a, s, t)$ as $s$ tends to infinity. This problem is specificly evident when analyzing a function with a Fourier transform concentrated along the $\omega_2$-axis, this causes the shearlet coefficients to be associated with very large $s$. This issue is adressed by partitioning the Fourier domain into specific regions which limits $s$ to finite intervals. Therefore, we proceed by restraining us to band-limited shearlets only, since this gives compact support in the Fourier domain. We construct a partition of the Fourier domain as the *cones* defined by

$$\begin{cases} \mathcal{C}^h \coloneqq \{(\omega_1, \omega_2) \in \mathbb{R}^2 : |\omega_1| \geq \frac{1}{2}, |\omega_2| < |\omega_1|\}, \\ \mathcal{C}^v \coloneqq \{(\omega_1, \omega_2) \in \mathbb{R}^2 : |\omega_2| \geq \frac{1}{2}, |\omega_2| > |\omega_1|\}, \\ \mathcal{C}^\times \coloneqq \{(\omega_1, \omega_2) \in \mathbb{R}^2 : |\omega_1| \geq \frac{1}{2}, |\omega_2| \geq \frac{1}{2}, |\omega_1| = |\omega_2|\}, \\ \mathcal{C}^0 \coloneqq \{(\omega_1, \omega_2) \in \mathbb{R}^2 : |\omega_1| < 1, |\omega_2| < 1\}. \end{cases}$$

We assign a shearlet to each region. Define the characteristic function $\chi_{\mathcal{C}^\kappa}(\omega)$ such that $\chi_{\mathcal{C}^\kappa}(\omega)$ is equal to 1 for $\omega \in \mathcal{C}^\kappa$ and zero elsewhere, where $\kappa \in \{v, h, \times\}$. Define the

[1]Original image by author [23].

*Cone-adapted shearlets* as

$$\begin{cases} \hat{\psi}^h(\omega_1,\omega_2) \coloneqq \hat{\psi}_1(\omega_1)\hat{\psi}_2\left(\frac{\omega_2}{\omega_1}\right)\chi_{\mathcal{C}^h}(\omega) & \omega \in \mathcal{C}^h\,, \\ \hat{\psi}^v(\omega_1,\omega_2) \coloneqq \hat{\psi}_1(\omega_2)\hat{\psi}_2\left(\frac{\omega_1}{\omega_2}\right)\chi_{\mathcal{C}^v}(\omega) & \omega \in \mathcal{C}^v\,, \\ \hat{\psi}^\times(\omega_1,\omega_2) \coloneqq \hat{\psi}_1(\omega_1)\hat{\psi}_2\left(\frac{\omega_2}{\omega_1}\right)\chi_{\mathcal{C}^\times}(\omega) & \omega \in \mathcal{C}^\times\,. \end{cases}$$

For the low-frequency part $\mathcal{C}^0$, we define a scaling function $\varphi$ by

$$\varphi(\omega) \coloneqq \begin{cases} 1 & |\omega| \le \frac{1}{2}\,, \\ \cos(\frac{\pi}{2}v(2|\omega|-1)) & \frac{1}{2} < |\omega| < 1\,, \\ 0 & \text{else}\,. \end{cases}$$

With $\varphi$ we construct a full scaling function $\phi$ by using $\varphi$

$$\hat{\phi}(\omega_1,\omega_2) \coloneqq \begin{cases} \varphi(\omega_1) & |\omega_2| \le |\omega_1|\,, \\ \varphi(\omega_2) & |\omega_1| < |\omega_2|\,, \end{cases}$$

which explicitly can be written as

$$\hat{\phi}(\omega_1,\omega_2) = \begin{cases} 1 & |\omega_1| \le \frac{1}{2}, |\omega_2| \le \frac{1}{2}\,, \\ \cos(\frac{\pi}{2}v(2|\omega_1|-1)) & \frac{1}{2} < |\omega_1| < 1, |\omega_2| \le |\omega_1|\,, \\ \cos(\frac{\pi}{2}v(2|\omega_2|-1)) & \frac{1}{2} < |\omega_2| < 1, |\omega_1| < |\omega_2|\,, \\ 0 & \text{else}\,. \end{cases} \tag{2.11}$$

The decay of $\hat{\phi}$ is chosen to exactly match the increase of $\hat{\psi}_1$, consequently, $\phi$ satisfies for $|\omega| \in [\frac{1}{2}, 1]$

$$|\hat{\psi}_1(\omega)|^2 + |\varphi(\omega)|^2 = \sin^2\left(\frac{\pi}{2}v(2|\omega|-1)\right) + \cos^2\left(\frac{\pi}{2}v(2|\omega|-1)\right) = 1\,.$$

### 2.3.4 Finite discrete shearlets

The question is how we can discretize the parameters $a$ and $s$ such that the compact support of $\hat{\psi}^h_{a,s,0}$ is a subset of the horizontal cone (and analogously for $\hat{\psi}^v_{a,s,0}$ and the vertical cone). Meanwhile we also need to discretize the parameters such that the discrete system still constitutes a frame. We do not pursue the full derivation of the range of the discrete parameters. However very briefly, by analyzing the support of $\hat{\psi}_{a,s,0}$, namely

$$\operatorname{supp}(\hat{\psi}_{a,s,0}) \subseteq \left\{(\omega_1,\omega_2) : \frac{1}{2a} \le |\omega_1| \le \frac{4}{a}\,,\ \left|s + \frac{\omega_2}{\omega_1}\right| \le \sqrt{a}\right\}\,,$$

one can derive the restrictions $|a| \le 1$ and $|s| \le 1$. Then consider an image of size $M \times N$, the restrictions of $a$ and $s$ gives us *the number of considered scales* $j_0 \coloneqq$

$\lfloor \frac{1}{2} \log_2 \max(M, N) \rfloor$. Thus, discretizing the dilation, shearing and translation parameters as

$$\begin{aligned} a_j &\coloneqq 2^{-2j} = 1/4^j \quad && j = 0, ..., j_0 - 1\,, \\ s_{j,k} &\coloneqq k2^{-j} && |k| \leq 2^j\,, \\ t_m &\coloneqq \left(\frac{m_1}{M}, \frac{m_2}{N}\right) && m_1 = 0, ..., M-1, m_2 = 0, ..., N-1\,. \end{aligned}$$

Using the discretized parameters we can rewrite our shearlet as

$$\psi_{j,k,m}(x) \coloneqq \psi(A^{-1}_{a_j, \frac{1}{2}} S^{-1}_{s_j,k}(x - t_m))\,.$$

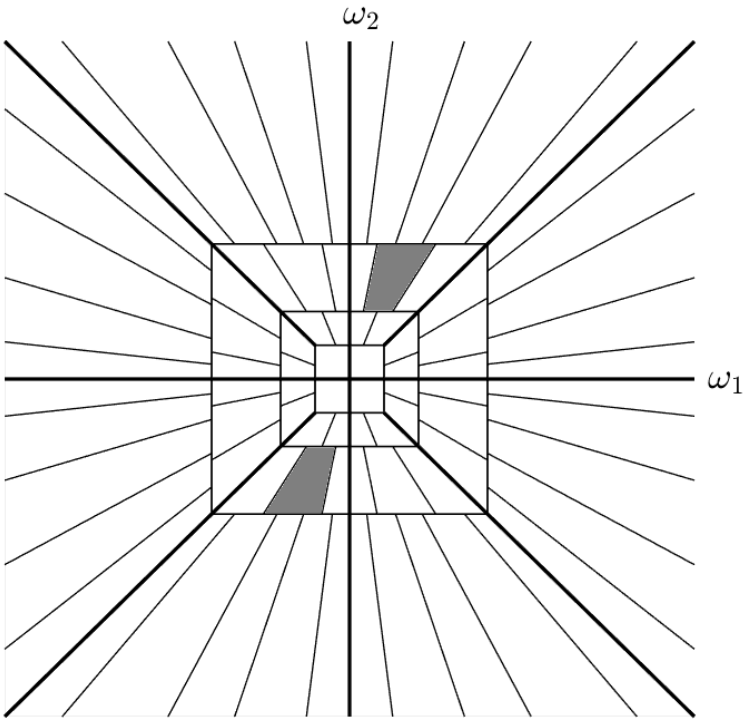


**Figure 2.8:** The induced shearlets have support in the Fourier domain in a similar pattern as indicated by the figure above. For a certain pair of parameters we obtain support on the grey areas. When changing $k$ we move through each trapezoidal region in any of the squares determined by $j$. For example $k = 0$ corresponds to the horizontal and vertical trapezoids, $k = \pm 2^j$ corresponds to the diagonal trapezoids, and any other line to any integer between $-2^j + 1$ and $2^j - 1$.

We have in the Fourier domain

$$\hat{\psi}_{j,k,m}(\omega) = a^{-\frac{3}{4}} \hat{\psi}_1(4^{-j}\omega_1) \hat{\psi}_2 \left(2^j \frac{\omega_2}{\omega_1} + k\right) e^{-2\pi i \left\langle \omega, \binom{m_1/M}{m_2/N} \right\rangle}\,.$$

In the Fourier domain $\Omega \subseteq \mathbb{Z}^2$ we use the coordinates $(\omega_1, \omega_2) \in \Omega$ such that $\omega_1 \in \{-\lfloor \frac{M}{2} \rfloor, ..., \lceil \frac{M}{2} \rceil - 1\}$ and $\omega_2 \in \{-\lfloor \frac{N}{2} \rfloor, ..., \lceil \frac{N}{2} \rceil - 1\}$. With this setup we have a cut off on the boundaries where $|k| = 2^j$. Thus $|s| = |2^j 2^{-j}| = 1$ which results in half shearlets around each side of $\mathcal{C}^\times$. Therefore, for $|k| = 2^j$, we define

$$\hat{\psi}^{h\times v}_{j,k,m} \coloneqq \hat{\psi}^{h}_{j,k,m} + \hat{\psi}^{v}_{j,k,m} + \hat{\psi}^{\times}_{j,k,m}\,.$$

We let $\phi_m = \phi(\cdot - m)$. Conclusively this gives us the discrete shearlet transform calculated as

$$\mathscr{SH}(f)(\kappa, j, k, m) \coloneqq \begin{cases} \langle f, \phi_m \rangle & \kappa = 0\,, \\ \langle f, \psi^{\kappa}_{j,k,m} \rangle & \kappa \in \{h, v\}, k \in \{-2^j + 1, ..., 2^j - 1\}\,, \\ \langle f, \psi^{h\times v}_{j,k,m} \rangle & \kappa = \times, |k| = 2^j\,. \end{cases}$$

Hence, the shearlet transform is practically realized by the following computations:

$$\mathscr{SH}(f) = \begin{cases} \texttt{ifft2}(\hat{\phi}(\omega_1, \omega_2)\hat{f}(\omega_1, \omega_2)) & \text{for } \kappa = 0\,, \\ \texttt{ifft2}(\hat{\psi}^h(4^{-j}\omega_1, 4^{-j}k\omega_1 + 2^{-j}\omega_2)\hat{f}(\omega_1, \omega_2)) & \text{for } \kappa = h, |k| \leq 2^j - 1\,, \\ \texttt{ifft2}(\hat{\psi}^v(4^{-j}\omega_2, 4^{-j}k\omega_2 + 2^{-j}\omega_1)\hat{f}(\omega_1, \omega_2)) & \text{for } \kappa = v, |k| \leq 2^j - 1\,, \\ \texttt{ifft2}(\hat{\psi}^{h\times v}(4^{-j}\omega_1, 4^{-j}k\omega_1 + 2^{-j}\omega_2)\hat{f}(\omega_1, \omega_2)) & \text{for } \kappa \neq 0, |k| = 2^j\,. \end{cases}$$

Over a two-dimensional grid of size $256 \times 256$ we plot the basis functions supplied by the shearlet framework, these can be seen from a two-dimensional view in Figure 2.9 and a three-dimensional view in Figure 2.10. Recall that $j = 0, 1, ..., j_0 - 1$ and therefore the calculations in Equation (2.3.4) has to be performed for $j = 0, 1, ..., j_0 - 1$. Each value of $j$ gives $2^{j+2}$ matrices of shearlet coefficients, which in total is $1 + \sum_{j=0}^{j_0-1} 2^{j+2}$ matrices.

In a similar manner, if given the shearlet coefficients and we wish to find the original image $f$, we do this by the inverse calculations. That is, by applying `ifft2` to the following expression:

$$\begin{aligned} \hat{f}(\omega_1, \omega_2) = {} & \texttt{fft2}(c(0, \cdot))\hat{\phi}(\omega_1, \omega_2) \\ & + \sum_{j=0}^{j_0-1} \sum_{k=-2^j+1}^{2^j-1} \texttt{fft2}(c(h, j, k, \cdot))\hat{\psi}^h(4^{-j}\omega_1, 4^{-j}k\omega_1 + 2^{-j}\omega_2) \\ & + \sum_{j=0}^{j_0-1} \sum_{k=-2^j+1}^{2^j-1} \texttt{fft2}(c(v, j, k, \cdot))\hat{\psi}^v(4^{-j}\omega_2, 4^{-j}k\omega_2 + 2^{-j}\omega_1) \\ & + \sum_{j=0}^{j_0-1} \sum_{k=\pm 2^j} \texttt{fft2}(c(h \times v, j, k, \cdot))\hat{\psi}^{h\times v}(4^{-j}\omega_1, 4^{-j}k\omega_1 + 2^{-j}\omega_2)\,. \end{aligned}$$

Then the original image is obtained by inverse transforming $\hat{f}$. This concludes the implementation of the shearlet transform. The discussed method is not the only method available to date. Another popular approach is based upon the pseudo-polar Fourier transform. For the interested reader we refer to [24].

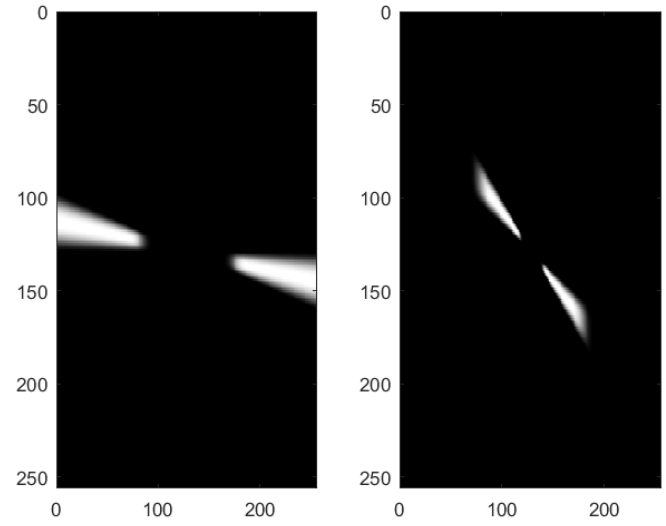


**Figure 2.9:** This figure illustrates shearlet basis functions in the Fourier domain from a two-dimensional view for two pairs of values $(j, k)$.

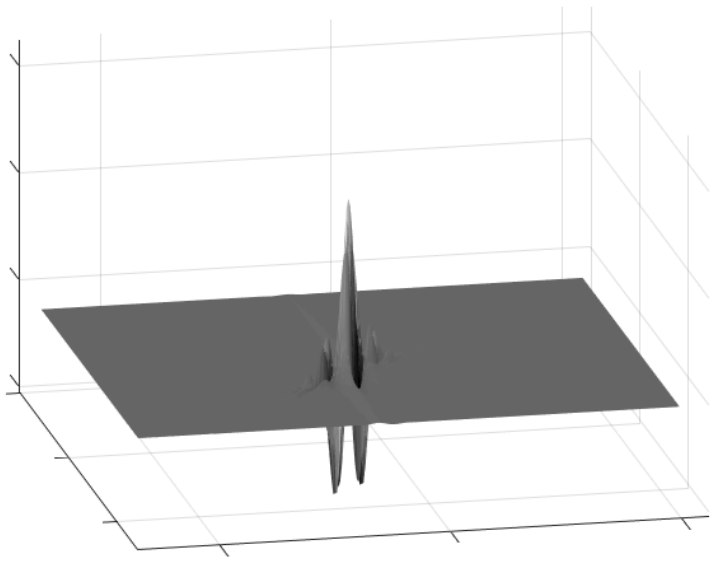

**Figure 2.10:** The shearlet basis in the time-domain for some parameters $(j, k)$.

# Chapter 3

# Sparsity and structure of shearlet coefficients

In this chapter we analyze the sparsity provided by the shearlet coefficients and illustrate how the structure of the shearlet transform can be applied to find edges with certain orientations. First we compare the shearlet coefficients with wavelet coefficients using a Daubechie-4 basis and looking at the best $N$-term approximations. Then finally we look at how the structure of the shearlet coefficients can be utilized to filter out features with certain sizes and orientations.

## 3.1 The N-term approximation

We remind the reader of the asymptotic decay we remarked upon in chapter 2.3, namely: Denote $f_N$ as the shearlet approximation of a *cartoon-like* image $f$ by using the $N$ largest shearlet coefficients, the error between $f$ and $f_N$ satisfies the following decay rate [21]:

$$||f - f_N||^2_{L^2(\mathbb{R}^2)} \leq CN^{-2}(\log N)^3\,, \qquad N \to \infty\,,$$

for some $C > 0$. If we do a similar comparison of approximating $f$ by $f_N$ using the best $N-$term approximation of wavelet coefficients, we have the following decay rate

$$\|f - f_N\|^2_{L^2(\mathbb{R}^2)} \leq CN^{-1}\,, \qquad N \to \infty\,.$$

We begin by briefly looking into these error rates. In this chapter we consider five images. The first one is an image of a leaf shown in 3.1 and the other images are textures shown in 3.2.

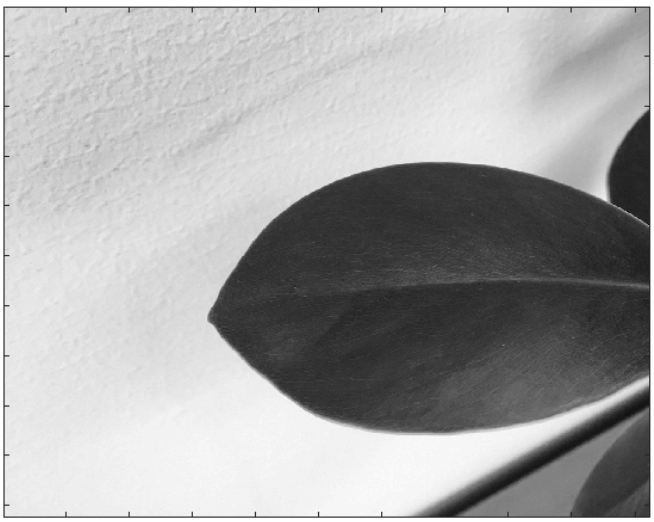

**Figure 3.1:** The image with a leaf.

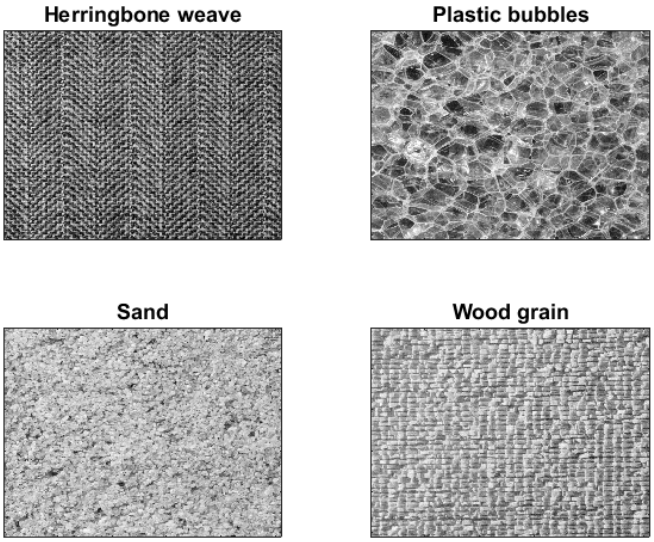


**Figure 3.2:** This figure contains the four different types of images we consider in this chapter. Each image represents a texture.

All images used are of size $1024 \times 1024$. The image of Figure 3.1 is taken by ourselves.

The texture images in Figure 3.2 are samples from the Brodatz dataset[1].

To analyze how $f_N$ relates to $f$ we use two approaches. The first is truncating all transform coefficients except the $N$ largest cofficients. The second approach considers sorting the coefficients in a descending order and then preserving $p\%$ of the largest coefficients, thus $100 - p\%$ coefficients are set to zero.

We begin by calculating the error rates $\|f - f_N\|^2$ using $1024^2$ of the largest coefficients for the the image leaf and we obtain the results shown in Figure 3.3. We also tried this for $N = 10^2, 10^3$ but the results are similar.

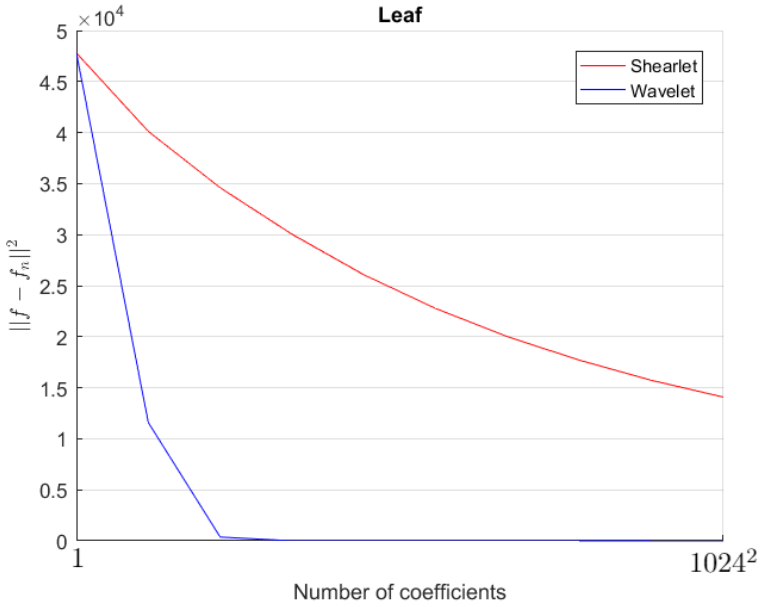


**Figure 3.3:** This figure contains the error rates using the $1024^2$ of the largest coefficients from each transform.

From the image it looks like the wavelet approximation is outperforming the shearlet approximation. However note that the images are of size $1024 \times 1024$ and therefore we have $c_w = 1024^2$ wavelet coefficients and $c_\psi = 1024^2 \times 125$ shearlet coefficients. Thus there are 125 times more coefficients from the shearlet transform. We continue by calculating $\|f - f_N\|^2$ again but this time analyzing the error by using a percentage of coefficients from each transform. If we plot the error using 1% to 95% of the coefficients from each transform, we obtain the results shown in Figure 3.4.

[1] `http://sipi.usc.edu/database/`

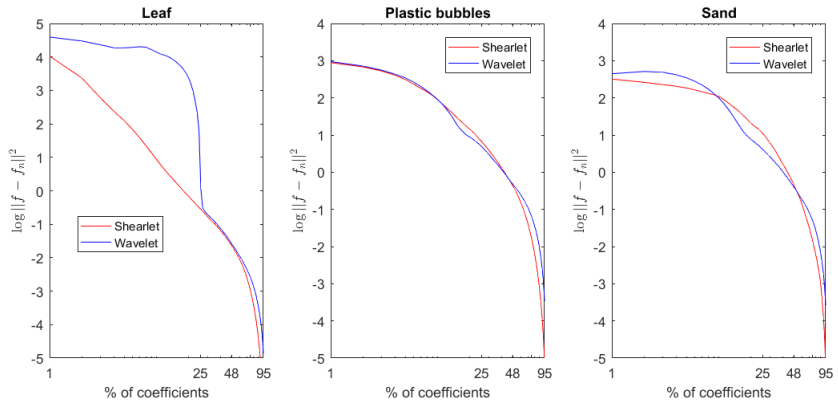


**Figure 3.4:** This figure contains the error rate using 1.00% to 95.00% of the coefficients from each transform. The red line is the shearlet approximation, the blue one the wavelet approximation. The scale in the $x$-direction is logarithmic. Note that there is a clear difference between each image.

It is interesting to visually see the effect of truncating the coefficients, i.e., see the reprodued images after preserving $p$% of the largest coefficients. We do this for $p \approx 1\%, 7.3\%, 13.6\%, 20\%$. We do not choose larger values of $p$ because after 20% it is difficult to tell any difference between the reproduced and original image. Applying this to the leaf image we obtain the results shown in 3.5 and 3.6. Note how the shearlet approximation captures the real image well at 7.3% compared to the wavelet approximation, which has removed several important features. Moreover at 20% it is difficult to tell any difference between the original image and the shearlet approximation, however the wavelet approximation removed several details from the background of the image. The edge of the leaf represents to a high degree a cartoon-like feature which is also probably the reason why the shearlets handle the image to a much greater extent compared to wavelets.

We also show the result for the images plastic bubbles in Figure 3.7, 3.8, for herringbone weave in Figure 3.9 and 3.10. Most notably we see that the reconstructed images using the shearlet transform, filters the image and preserves certain edges. If we compare the truncation results with the wavelet equivalent process we see (as expected) more isotropic features preserved. In the more anisotropic texture (herringbone weave) we see that 1% of the largest shearlet coefficients preserve the image well due to its richness in anisotropic features.

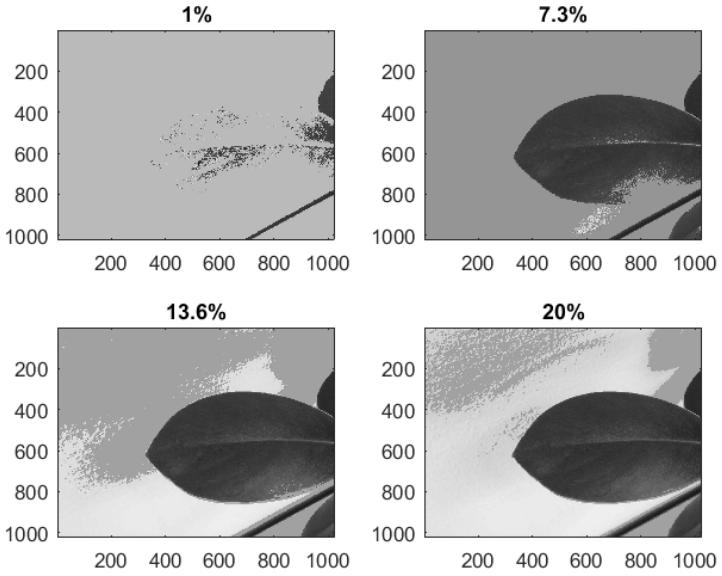


**Figure 3.5:** Using the image leaf, this figure shows the result of preserving $p\%$ of the largest wavelet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

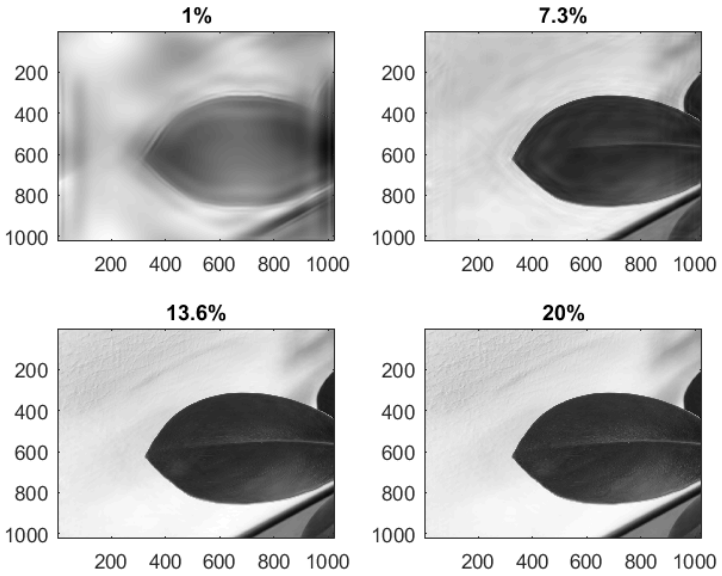


**Figure 3.6:** Using the image leaf, this figure shows the result of preserving $p\%$ of the largest shearlet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

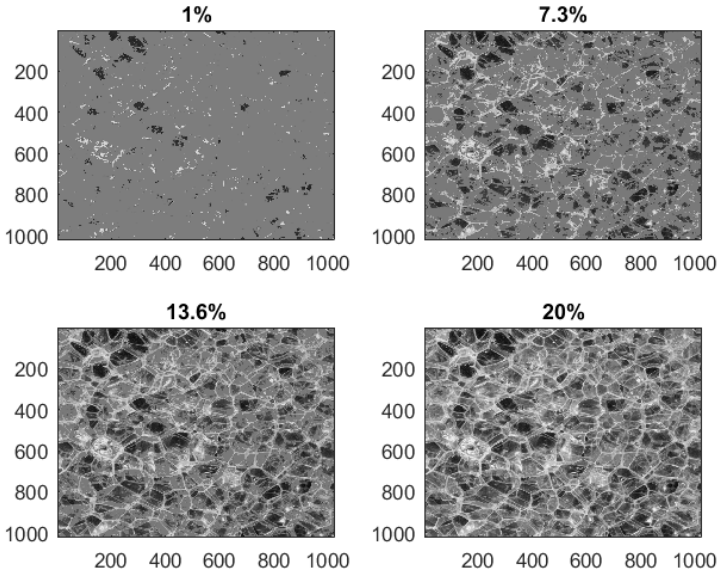


**Figure 3.7:** Using the image plastic bubbles, this figure shows the result of preserving $p\%$ of the largest wavelet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

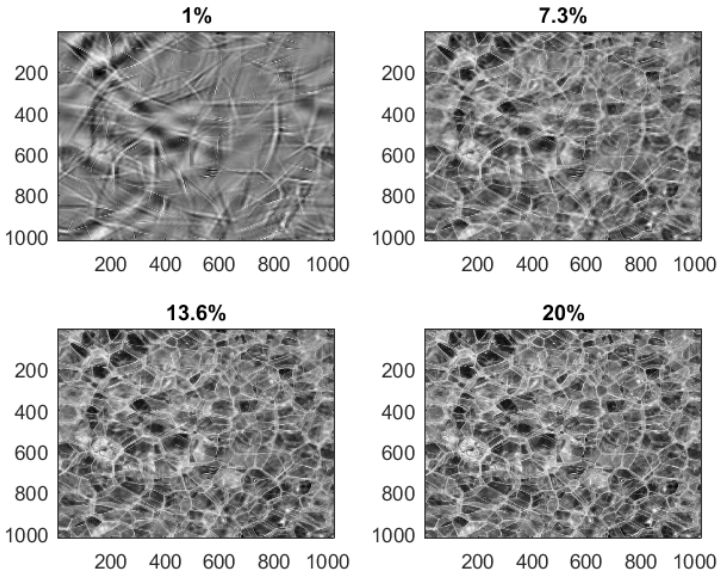


**Figure 3.8:** Using the image plastic bubbles, this figure shows the result of preserving $p\%$ of the largest shearlet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

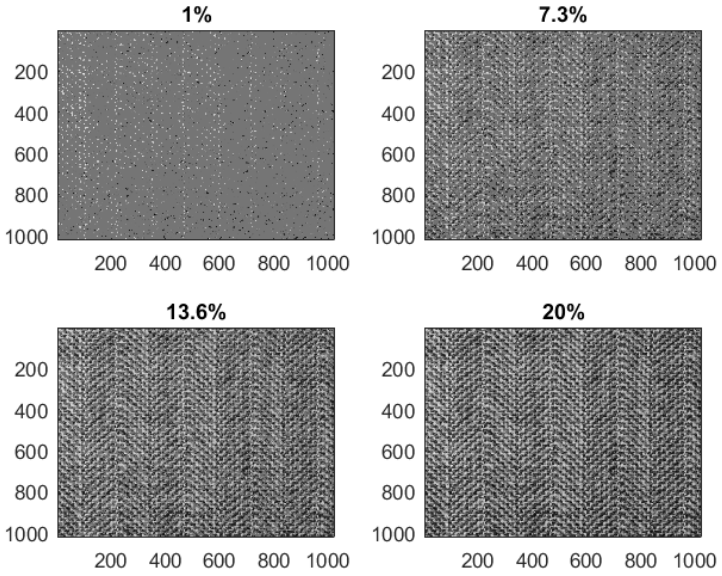


**Figure 3.9:** Using the image herringbone weave, this figure shows the result of preserving $p\%$ of the largest wavelet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

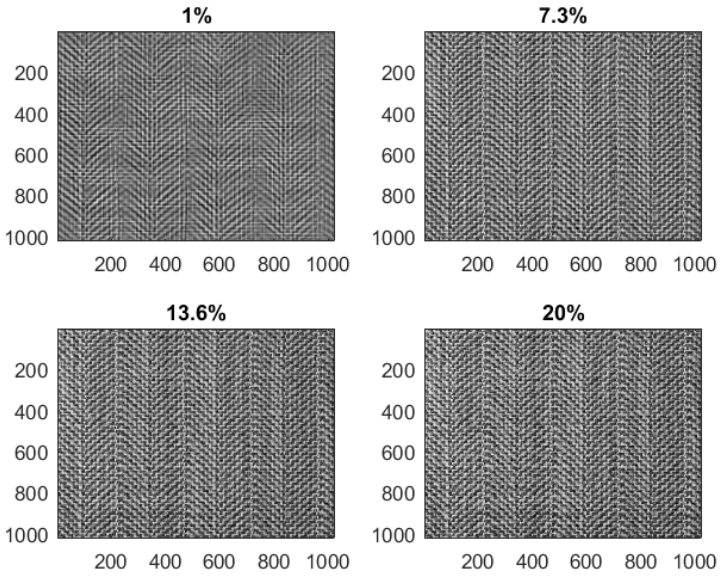


**Figure 3.10:** Using the image herringbone weave, this figure shows the result of preserving $p\%$ of the largest shearlet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

Finally for sand and wood grain we obtain the results shown in Figures 3.11, 3.12, 3.13 and 3.14. Note that even though the image of sand is a very isotropic image, still edges are detected when truncating a large amount of the shearlet coefficients. However the wavelet representation appears to be more like the original image due to the lack of anisotropic features. For the wood grain images we see again that the shearlet images represent the original image to a very high degree due to its anisotropic richness.

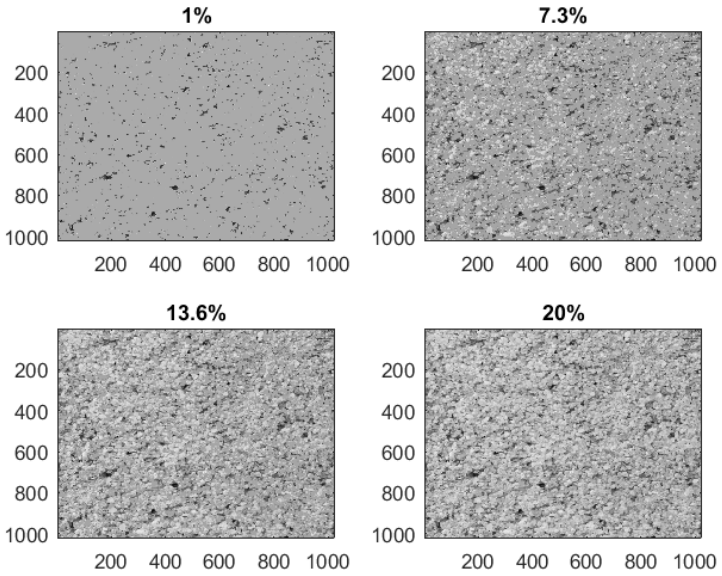


**Figure 3.11:** Using the image sand, this figure shows the result of preserving $p\%$ of the largest wavelet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

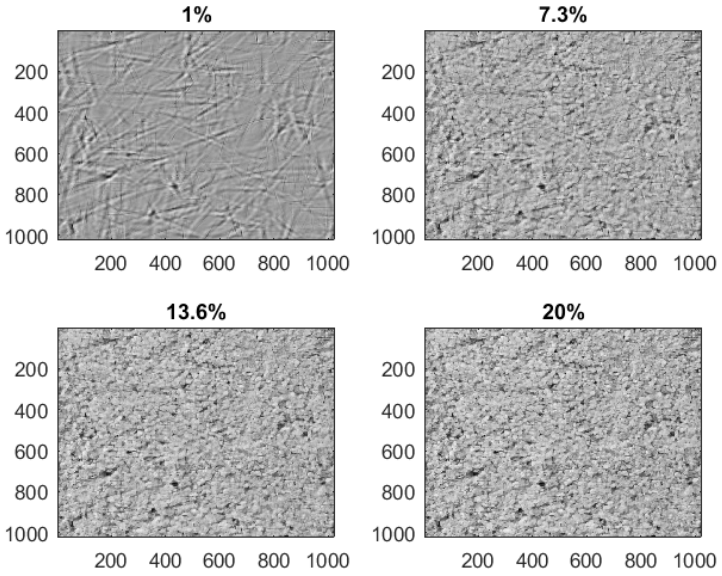


**Figure 3.12:** Using the image sand, this figure shows the result of preserving $p\%$ of the largest shearlet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

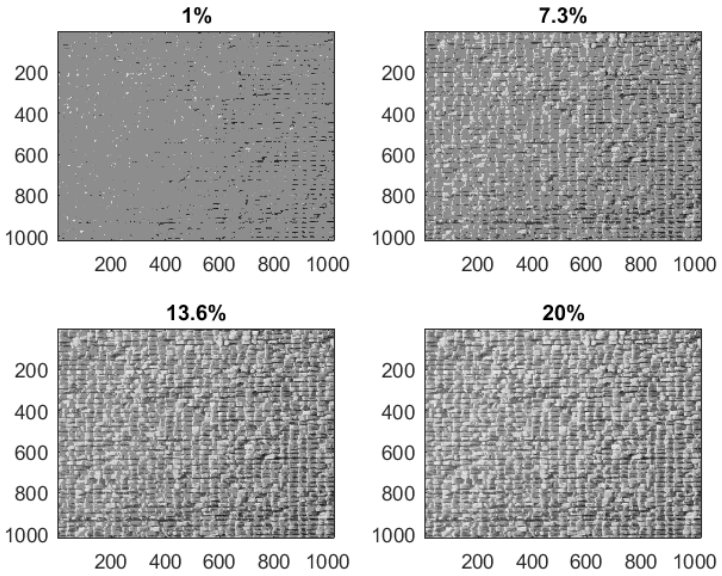


**Figure 3.13:** Using the image wood grain, this figure shows the result of preserving $p\%$ of the largest wavelet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

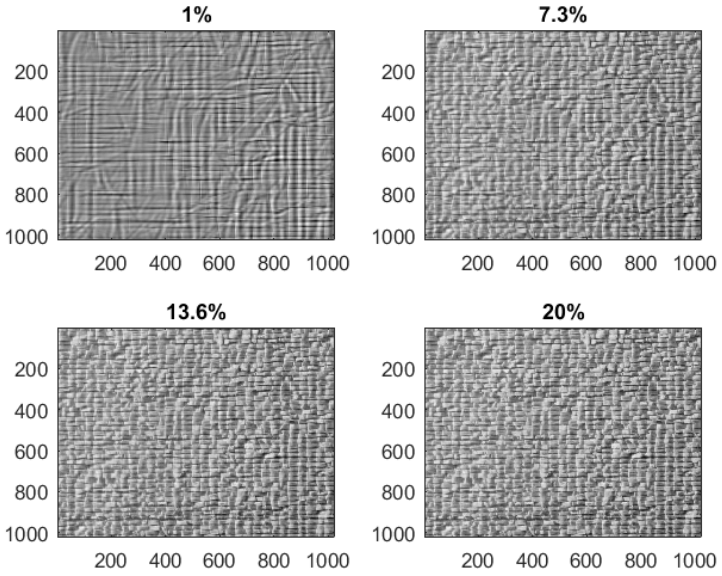


**Figure 3.14:** Using the image wood grain, this figure shows the result of preserving $p\%$ of the largest shearlet coefficients and reproducing the truncated image. The value of $p$ is indicated by its respective title.

Note that due to the high amount of sharp edges the wood grain image, the original image is well resembled in Figure 3.14 using only 7% of the largest coefficients.

## 3.2 Filtering by utilizing the layers of shearlet coefficients

Finally we show how one can utilize the parameters $j$ and $k$ to filter out information of certain scales and directions. Recall that the parameter $j$ is limited by the size of the image. For the images of size $1024 \times 1024$ we consider five values of $j$. By inverse transforming coefficients related to certain values of $j$, we can extract details of specific sizes. The procedure is simple and is the following: Pick a value of $j$ and inverse transform only the coefficients related to that value of $j$. By doing this we filter out features of different sizes determined by the value of $j$. For instance for $j = 1$ or $j = 2$ we obtain information about very coarse scales. For $j = 4$ and $j = 5$ we obtain information for very fine scales. Doing this, we show the result for the leaf in Figure 3.15 and here we see that some values of $j$ keeps the edges of the leaf. The fine scale layers capture the details on the leaf and patterns in the background, which is also expected.

For the image plastic bubbles we show the results in Figure 3.16, here we see that the edges of the bubbles are more preserved for $j = 3$. For $j = 2$ we seem to have located large spots due to the variations in light in the original image. For the sand image we have the results in 3.17 which are very similar to the results from the plastic bubbles.

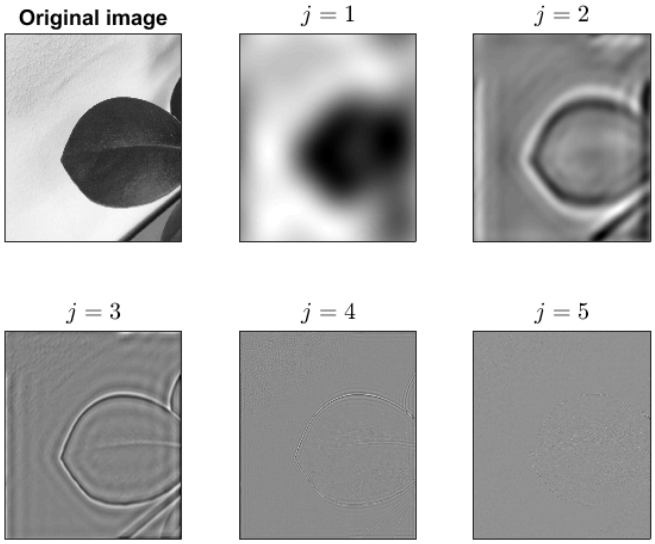


**Figure 3.15:** In the top left corner we see the original image. Then by inverse transforming for different values of $j$ we obtain the images that follows.

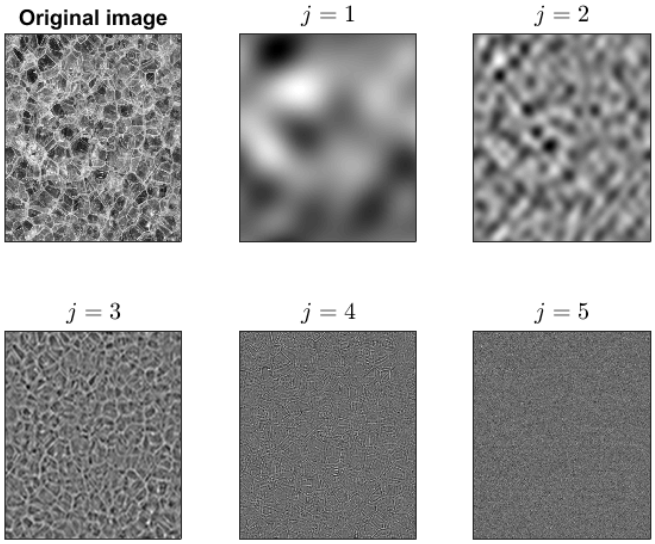


**Figure 3.16:** In the top left corner we see the original image. Then by inverse transforming for different values of $j$ we obtain the images that follows.

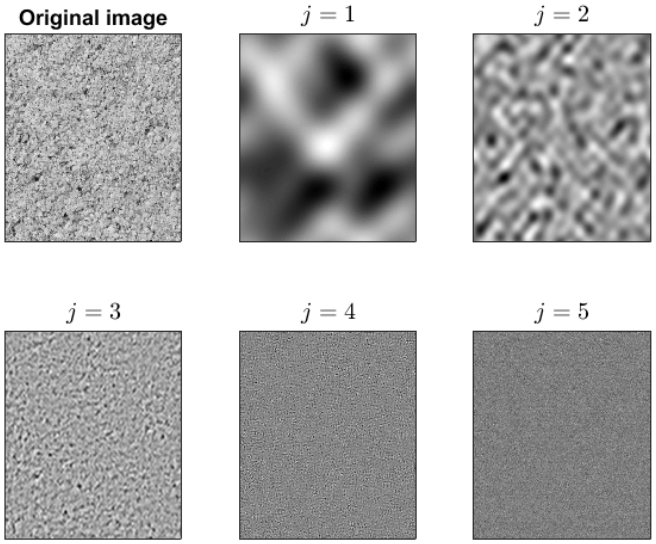


**Figure 3.17:** In the top left corner we see the original image. Then by inverse transforming for different values of $j$ we obtain the images that follows.

It is also interesting to compare the energy of each image in Figure 3.15, 3.16 and 3.17. The energy of each image is the sum of the squared transform coefficients for each $j$. We obtain the results shown in Figure 3.18. Note how the energies varies between each image. For instance almost all the energy in the leaf image are in the first three layers, while for plastic bubbles and sand the energy is more concentrated in the middle layers.

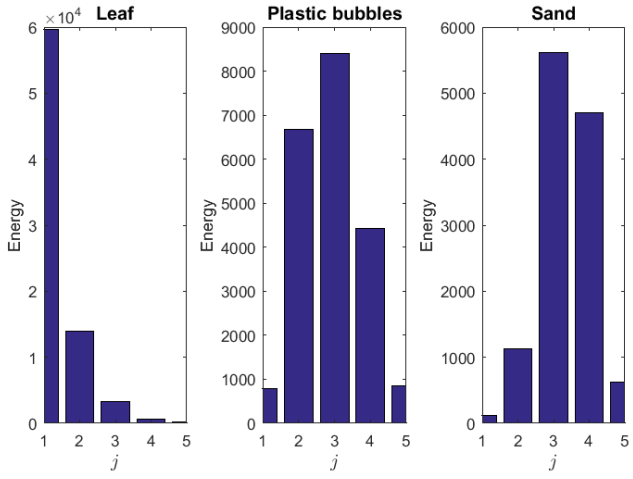


**Figure 3.18:** The energies of each layer $j = 1, ..., 5$ for the images leaf, plastic bubbles and sand.

With a similar approach we can inverse transform coefficients related to specific values of $k$. By doing this, we expect to locate details with specific orientations instead of sizes. We pursue this for the herringbone weave due to its diagonal structures, we capture details from two specific directions shown in Figure 3.19.

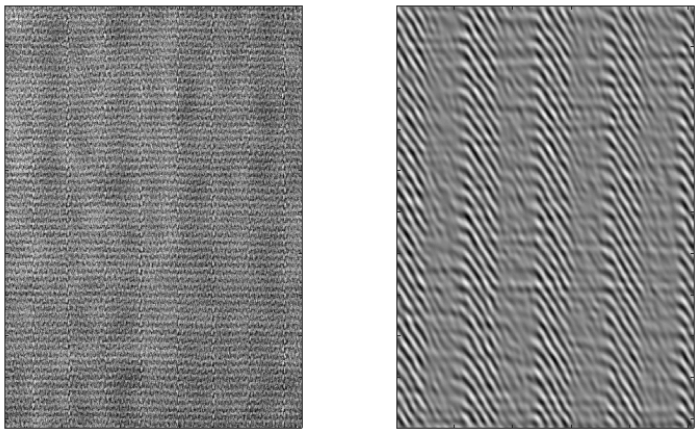

**Figure 3.19:** The original image is the herringbone weave image. The left image is a reconstruction of horizontal details by choosing $j$ and $k$ carefully. In the leftmost image we set all coefficients for $j = 2$ to zero. Due to the thickness of the diagonal lines in the original image, these are not captured in other layers. In the rightmost image we set all coefficients to zero except for those related to $j = 2$ and $k$ for the first and third quadrant, which preserves half of the directional information in that layer.

By considering the coarsest scale we capture the variations in the overall brightness in an image. Thus the features provided by the shearlet coefficients offer several interesting applications which is related to localizing certain details, related to both its size and orientation. For instance in Figure 3.16 we saw how we can locate dark spots in an image. In Figure 3.6 we located details of certain sizes such as the edge of the leaf or spots on the wall. In Figure 3.19 we located features with specific orientations.

The framework of shearlets offer several interesting approaches to filter images to locate specific features. There are of course a numerous of applications of this to filter images. However perhaps one of the most interesting applications is to apply this in medical sciences to possibly aid in detection of cancer. This would be an interesting approach for future work.

# Chapter 4

# Support vector machine

This chapter is about a method for data classification, called *support vector machine* (SVM) [25]. It is a non-probabilistic binary classifier that ideally splits data into two classes by a hyperplane. When data is not linearly separable, i.e. when there is no hyperplane separating the two classes, SVM is extended using a so called kernel trick. For problems with more than two classes, combinations of binary SVM are used to obtain a multiclass classifier.

In the later part of this chapter we briefly outline a metric for the non-linear space of symmetric positive definite marices and descibe how this can be implemented to improve the classification rate of a model.

## 4.1 Support vector machine

A well known model for data classification is the support vector machine. The SVM model is usually introduced as a binary classifier. Multiclass extensions exist and are based on combining binary classifiers. We start with the binary case. The aim of using a binary SVM is to be able to tell a difference (if it exists) between two different classes of data. Sometimes it is difficult to see any difference from two different classes of data. If a set of data is difficult to separate into individual classes, there exist different methods that ideally makes it easier to separate. One popular method is *the kernel trick* which is described later in this section.

We begin by emphasizing some basic feautures of SVMs. In a SVM model we wish to find a hyperplane with normal vector $\boldsymbol{w} \in \mathbb{R}^m$ such that the minimal distance between each data point $\boldsymbol{x}_i$ and the hyperplane is maximized under certain constraints. If $\boldsymbol{w}$ denotes the normal of the hyperplane, and $b$ denotes the offset of the hyperplane from the origin, we can write a hyperplane in $\mathbb{R}^m$ as the set of points $\boldsymbol{x} \in \mathbb{R}^m$ that satisfies $\boldsymbol{w} \cdot \boldsymbol{x} = b$. If it is possible to separate the data by a hyperplane we can separate the data by two additional parallel hyperplanes known as *margins*. These margins are either categorized as *hard margins* or *soft margins*. Soft margins are preferred when it is not possible to separate all the data, but a majority of it by introducing a cost-function that penalizes data on wrong side of the margins. Hard margins allow zero classification error

compared to soft margins. However the zero error property with hard margins does not come for free. Hard margins can result in an overfitting model, while also being very sensitive to noise. The margins satisfies $\boldsymbol{w} \cdot \boldsymbol{x} - b = \pm 1$. Data located on the margins are known as *support vectors*. To illustrate a scenario, an example with arbitrary data in $\mathbb{R}^2$ can be seen in Figure 4.1.

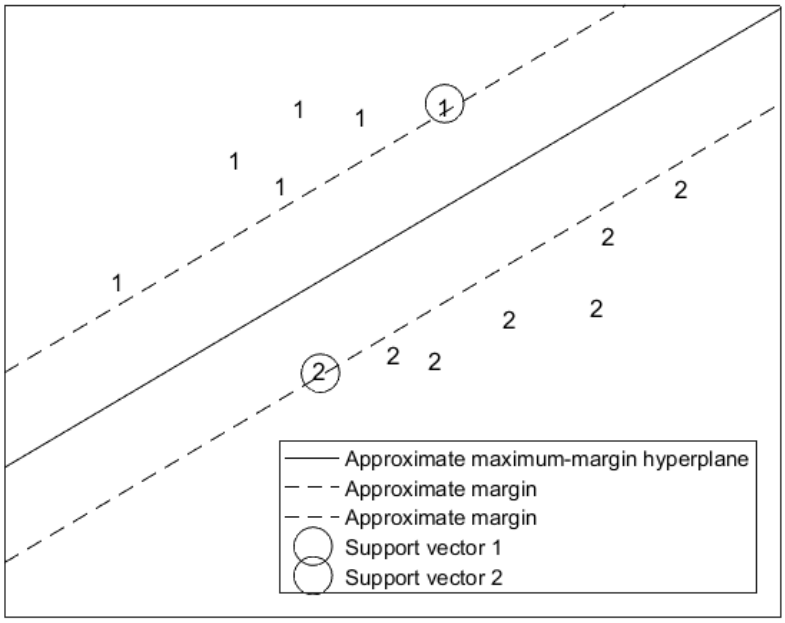


**Figure 4.1:** This figure illustrates data from two different groups indicated by numbers 1 and 2. In this figure we see a hard-margin SVM and the margins are indicated by dashed line (only approximate, as an illustrative example). The data marked with circles are called *support vectors*.

We continue with introducing the SVM. Consider data $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{C}^m$ with corresponding *class labels* $y_1, ..., y_n \in \{-1, 1\}$. Here $y_n = 1$ denotes the class with corresponding label "1", and $y_n = -1$ the class with corresponding label "-1". As mentioned earlier, we wish to find a hyperplane with normal $\boldsymbol{w}$ that divides the two data groups with a maximal distance. In other words, we want to maximize the distance between the margins which are separated by a distance $2/\|\boldsymbol{w}\|$. This is equivalent to minimizing $\|\boldsymbol{w}\|$. Moreover using hard margins we require each $\boldsymbol{x}_i$ to be located at the "correct" side of the hyperplane, hence we minimize $\|\boldsymbol{w}\|$ under the constraints $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$, for each $i \in \{1, ..., n\}$. Solving the minimization problem gives $\boldsymbol{w}$ and $b$. New data $\boldsymbol{x}_{\text{new}}$ is classified by $\text{sign}(\boldsymbol{w} \cdot \boldsymbol{x}_{\text{new}} - b)$. The output is equal to $\pm 1$ depending on which side of the hyperplane $\boldsymbol{x}_{\text{new}}$ is located at.

If one wishes to use a soft margin instead of a hard margin the objective function $\|\boldsymbol{w}\|$ is replaced by

$$\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b)) + \lambda\|\boldsymbol{w}\|^2\,,$$

under the constraints $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \max(0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b))$. The parameter $\lambda$ determines trade-off between having data $\boldsymbol{x}_i$ on the correct side of the margin and increasing margin-magnitude. We summarize the optimization problem below.

Consider $n$ data points and denote $\xi_i = \max(0, 1-y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b))$. The *primal* problem with a soft margin corresponds to the optimization problem

$$\begin{aligned} &\text{minimize} && \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\xi_i + \lambda\|\boldsymbol{w}\|^2\,, \\ &\text{subject to} && y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i\,, \quad \xi_i \geq 0\,,\ i = 1, ..., N\,. \end{aligned} \tag{4.1}$$

The objective function and the constraints are convex. This makes the optimization problem a convex one thus implying that the duality gap between the primal and dual problem is zero. When the duality gap is zero the primal and dual problem have equal optimal values. We derive the dual problem below.

We define the Lagrangian using the soft-margin model from (4.1) by multiplying the objective function with a factor $\frac{1}{2\lambda}$, which gives

$$\mathcal{L}(\boldsymbol{w}, b, \xi, c, r) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2\lambda n}\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}c_i\{y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) - 1 + \xi_i\} - \sum_{i=1}^{n}r_i\xi_i\,.$$

The terms $r_i\xi_i$ arise because we have the constraints $\xi_i \geq 0$. Denote the optimal solution to (4.1) by $p^*$. Solving the original primal problem is equivalent to computing

$$p^* = \min_{\boldsymbol{w},\, b,\, \xi}\ \max_{c\geq 0,\, r\geq 0}\mathcal{L}(\boldsymbol{w}, b, \xi, c, r)\,.$$

By strong duality, the optimal primal solution is equal to the optimal dual solution $d^*$, therefore

$$p^* = d^* = \max_{c\geq 0,\, r\geq 0}\ \min_{\boldsymbol{w},\, b,\, \xi}\mathcal{L}(\boldsymbol{w}, b, \xi, c, r)\,.$$

We solve equations corresponding to the partial derivatives of $\mathcal{L}$ with respect to $\boldsymbol{w}$, $b$ and $\xi$ equal to zero, i.e.

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = \frac{\partial\mathcal{L}}{\partial b} = \frac{\partial\mathcal{L}}{\partial\xi} = 0\,.$$

This gives

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = 0 \quad\Longleftrightarrow\quad \boldsymbol{w} - \sum_{i=1}^{n}c_i y_i\boldsymbol{x}_i = 0 \quad\Longleftrightarrow\quad \boldsymbol{w} = \sum_{i=1}^{n}c_i y_i\boldsymbol{x}_i\,.$$

$$\frac{\partial\mathcal{L}}{\partial b} = 0 \quad\Longleftrightarrow\quad \sum_{i=1}^{n}c_i y_i = 0\,.$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Leftrightarrow \frac{1}{2n\lambda} - c_i = r_i\,, \qquad i = 1, ..., n\,.$$

The last equality also implies $c_i \leq \frac{1}{2n\lambda}$ since $r_i \geq 0$. We continue by substituting the expressions from the partial derivatives into the Lagrangian. We obtain

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \xi, c, r) =& \frac{1}{2}\sum_{i,j=1}^{n} c_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) y_j c_j - \sum_{i=1}^{n} c_i \Big[ y_i \Big( \Big( \sum_{j=1}^{n} c_j y_j \boldsymbol{x}_j \Big) \cdot \boldsymbol{x}_i + b \Big) \Big] + \sum_{i=1}^{n} c_i \\ =& \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) c_j y_j\,.\end{aligned}$$

Therefore we obtain the dual problem for (4.1) as

$$\begin{array}{ll} \text{maximize} & f(c_1, ..., c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) y_j c_j\,, \\ \text{subject to} & \sum_{i=1}^{n} c_i y_i = 0\,, \\ & 0 \leq c_i \leq \frac{1}{2n\lambda}\,, \qquad i = 1, ..., n\,. \end{array} \tag{4.2}$$

Solving the dual problem gives a solution in terms of $c_i$ and the primal solution is related to the dual solution through $\boldsymbol{w} = \sum_{i=1}^{n} c_i y_i \boldsymbol{x}_i$. Finally the bias term $b$ is calculated using the support vectors, that is, using the non-zero weights $c_i$

$$b = \frac{1}{|\{i : 0 < c_i < \frac{1}{2n\lambda}\}|} \sum_{i:0<c_i<\frac{1}{2n\lambda}} \Big( y_i - \sum_{j:0<c_j<\frac{1}{2n\lambda}} c_j y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \Big)\,. \tag{4.3}$$

Thus we obtain a classifier given by the mapping

$$\boldsymbol{x} \mapsto \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x} - b)\,.$$

### 4.1.1 Kernels

Suppose that data $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{C}^m$ are not linearly separable in $\mathbb{C}^m$. This problem can be dealt with using some nonlinear function $\phi$ to map the data onto some space where it is linearly separable. This is often referred to as mapping data into a *feature space*, and $\phi$ represents the feature map. Unfortunately $\phi$ often maps data into a very high dimensional space causing an unacceptable increase in computational cost. The *kernel trick* is used to circumvent this problem. It is based on the fact that a kernel function $\mathcal{K} : \mathbb{C}^m \times \mathbb{C}^m \to \mathbb{R}$ satisfying the appropriate conditions, see below, implicitly induces a so called reproducing kernel Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}}, \| \cdot \|_{\mathcal{H}})$ and a map $\phi : \mathbb{C}^m \to \mathcal{H}$ satisfying $\mathcal{K}(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$, for $x, y \in \mathbb{C}^m$. For appropriate choices of a kernel $\mathcal{K}$ the induced feature space and map can make transformed data $\phi(x_1), ..., \phi(x_n)$ linearly separable in $\mathcal{H}$. As described below, $\mathcal{H}$ and $\phi$ are never used but scalar products $x \cdot y$, or rather $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ are replaced by $\mathcal{K}(x, y)$. This does not increase computational cost.

Mercer's Theorem states that if $\mathcal{K} : \mathbb{C}^m \times \mathbb{C}^m \to \mathbb{R}$ satisfies

$$\iint g(\alpha)\mathcal{K}(\alpha,\beta)g(\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta \geq 0\,,$$

for all $g \in L^2$, i.e. if $\mathcal{K}$ is a positive semidefinite kernel, then the space $\mathcal{H}$ and the function $\phi$ exist. Popular kernels are kernels based on polynomials and the radial basis kernel $\mathcal{K}(x,y) = \exp(-\gamma\|x-y\|^2)$.

To solve the optimization problem when using a kernel we consider the dual problem. The dual problem with a kernel $\mathcal{K}$ is

$$\begin{aligned} \text{maximize} \quad & f(c_1,...,c_n) = \textstyle\sum_{i=1}^n c_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y_i c_i \mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) y_j c_j\,, \\ \text{subject to} \quad & \textstyle\sum_{i=1}^n c_i y_i = 0\,, \\ & 0 \leq c_i \leq (2n\lambda)^{-1}\,, \qquad i = 1,...,n\,. \end{aligned} \tag{4.4}$$

The only difference from the problem without a kernel (4.2) is the scalar product, which has been replaced with the kernel. The bias term $b$ is calculated using the support vectors, i.e. the non-zero weights $c_i$ in the following way

$$b = \frac{1}{|\{i : 0 < c_i < \frac{1}{2n\lambda}\}|} \sum_{i:0<c_i<\frac{1}{2n\lambda}} \Big(y_i - \sum_{j:0<c_j<\frac{1}{2n\lambda}} c_j y_j \mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)\Big)\,. \tag{4.5}$$

The optimal hyperplane is a linear combination of the support vectors. This gives us the final mapping to classify new data, which is

$$\boldsymbol{x}_{\text{new}} \mapsto \operatorname{sign}\left(\sum_{i=1}^n c_i y_i \mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_{\text{new}}) + b\right)\,. \tag{4.6}$$

For the interest of the reader we derive the reproducing kernel Hilbert space $\mathcal{H}$ and the feature maps $\phi$ for the radial bassis kernel. Interestingly the space $\mathcal{H}$ is the infinite dimensional sequence space $\ell^2$ and thus completely infeasible for computations without kernel. By expanding $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \|\boldsymbol{x}_i\|^2 - 2\boldsymbol{x}_i^T\boldsymbol{x}_j + \|\boldsymbol{x}_j\|^2$ and then do a Taylor expansion we get

$$\begin{aligned} \mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) \\ &= \exp(-\gamma\|\boldsymbol{x}_i\|^2)\exp(-\gamma\|\boldsymbol{x}_j\|^2)\exp(2\gamma\boldsymbol{x}_i^T\boldsymbol{x}_j) \\ &= \exp(-\gamma\|\boldsymbol{x_i}\|^2)\exp(-\gamma\|\boldsymbol{x}_j\|^2)\sum_{k=0}^{\infty}\frac{(2\gamma\boldsymbol{x_i}^T\boldsymbol{x}_j)^k}{k!} \\ &= \exp(-\gamma\|\boldsymbol{x_i}\|^2)\exp(-\gamma\|\boldsymbol{x_j}\|^2))\sum_{k=0}^{\infty}\frac{(2\gamma)^k}{k!}(x_i^1 x_j^1 + ... + x_i^m x_j^m)^k\,, \end{aligned}$$

where $x_m^i$ denotes the $m$-th component of $\boldsymbol{x}_i$. This gives that the image $\phi(\boldsymbol{x})$ of $\boldsymbol{x} \in \mathbb{R}^m$

is explicitly given by

$$\phi(\boldsymbol{x}) = e^{-\gamma\|\boldsymbol{x}\|^2} \begin{pmatrix} 1 \\ \sqrt{2\gamma}x_1 \\ ... \\ \sqrt{2\gamma}x_m \\ \sqrt{2}\gamma^2 x_1^2 \\ \sqrt{2}\gamma^2 x_1 x_2 \\ ... \\ \sqrt{2}\gamma^2 x_1 x_m \\ \sqrt{2}\gamma^2 x_2 x_1 \\ \sqrt{2}\gamma^2 x_2^2 \\ ... \\ 2\gamma^2 x_2 x_m \\ ... \\ \sqrt{2}\gamma^2 x_m^2 \\ \sqrt{\frac{8}{6}\gamma^3} x_1^3 \\ ... \end{pmatrix},$$

and the reproducing kernel Hilbert space $\mathcal{H}$ is given by

$$\mathcal{H} = \ell^2 = \{(x_1, x_2, ...) : x_n \in \mathbb{C}, n \geq 1, \sum_{n=1}^{\infty} x_n^2 < \infty\}.$$

## 4.2 Multiclass support vector machines

The optimization problem for a multiclass support vector machine (MSVM) is identical to the binary case except that training is repeated between the classes and testing is performed differently. Training of a MSVM can be implemented in a number of different ways. There is no general agreement [26] on how testing and training should be done nor an argument that any of them is better than the other. In this section three different MSVM implementations are described, which are, *one-against-all*, *one-against-one* and the *directed acyclic graph support vector machines* (DAGSVM).

Consider data $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{C}^m$ from $k$ classes with corresponding labels $y_1, ..., y_n \in \{1, 2, ..., k\}$. The *one-against-all* model trains $k$ binary models where each of the $k$ models are treated once as data with corresponding label $+1$ while all the data from the other $k-1$ classes are treated as data with corresponding label $-1$. A figure illustrating this can be seen in Figure (4.2). Classification of data $\boldsymbol{x} \in \mathbb{R}^m$ is performed by

$$\underset{r=1,...,k}{\operatorname{argmax}} \sum_{i=1}^{n} c_i^r y_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b^r.$$

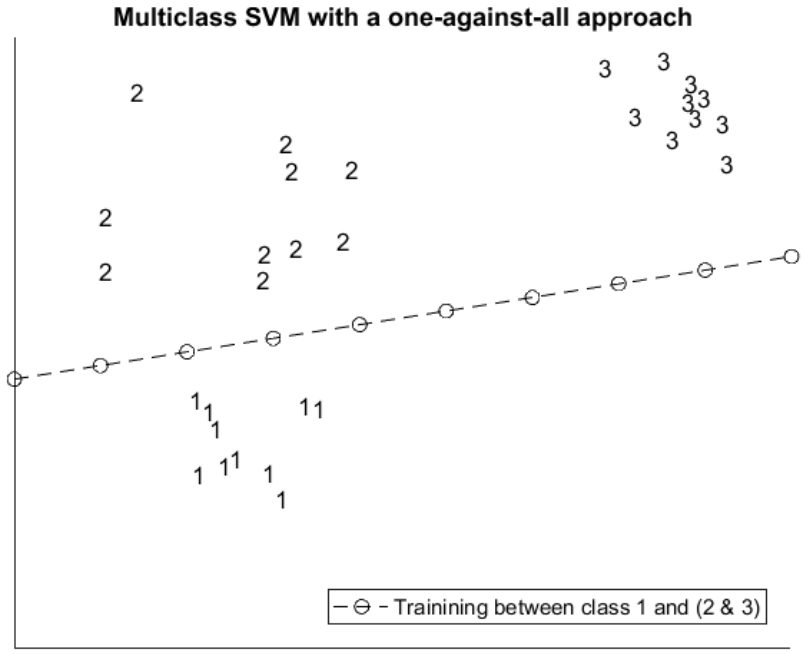


**Figure 4.2:** This figure illustrates a MSVM by using an one-against-all approach with a total of three different classes. Each number corresponds to a data point from the class with corresponding number. Only three classes are used for simplicity. Here the $m$th class ($1 \leq m \leq 3$) is assigned label $-1$ while the other classes are assigned class label $+1$. In this figure we illustrate how class 1 is compared to class 2 and 3. This process is then repeated by treating class 2 as a separate data set from 1 and 3, and finally by treating class 3 as a separate data set from 1 and 2.

A second approach is to consider *one-against-one*. Again, consider $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{C}^m$ from $k$ classes with corresponding labels $y_1, ..., y_n \in \{1, 2, ..., k\}$. This multiclass extension trains $k(k-1)/2$ classifiers instead of $k$ classifiers compared to the one-against-all model. Each classifier trains on data from two of the $k$ classes by considering them in pairs. A visual representation of this can be seen in Figure 4.3.

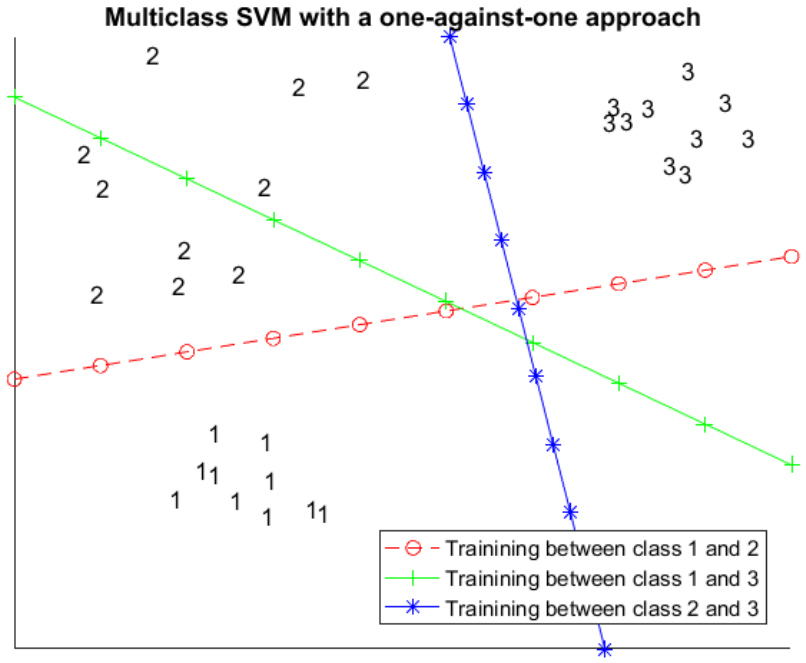


**Figure 4.3:** This figure illustrates the pairwise comparison that is performed with a one-against-one approach for a MSVM. Each number corresponds to a data point from the class with corresponding number. Each pair has to be trained only once. Training between class 1 and 2 is equivalent to train between class 2 and 1. In total we have $k(k-1)/2$ comparisons. In this illustrating example with three groups we have exactly 3 comparisons. The hyperplanes are approximate solutions to the data set and are drawn to illustrate the one-against-one scenario.

During classification using the one-against-one approach there exists no agreed upon method, however there are a few proposed ones and one is the "maximum amount of wins strategy". One classify data using all the $k(k-1)/2$ classifiers and labels $\boldsymbol{x}$ by the "winning class", that is, the class $\boldsymbol{x}$ was most frequently assigned to. If there are more classes than one with an equal and maximum amount of upvotes one assigns $\boldsymbol{x}$ to the class with lowest index.

The final approach we are to describe is the DAGSVM. The DAGSVM uses the same approach as the one-to-one classifier but classifies data differently. Data is classified using a tree-like structure that consists of nodes that corresponds to binary classifiers. With $k$ classes, data $\boldsymbol{x}$ is intially classified at the root node which corresponds to a classification between class $k$ and 1. This assigns $\boldsymbol{x}$ either to class $k$ or 1, and $\boldsymbol{x}$ moves therefore either left or right through the tree from its current node depending on the assignment. For instance assume $\boldsymbol{x}$ is assigned to $k$. Then $\boldsymbol{x}$ is classified at the next node corresponding to classes $k$ and 2. $\boldsymbol{x}$ is assigned to any of the classes and the process is repeated until it reaches a final label assignment. A graphical illustration of this tree-like structure can

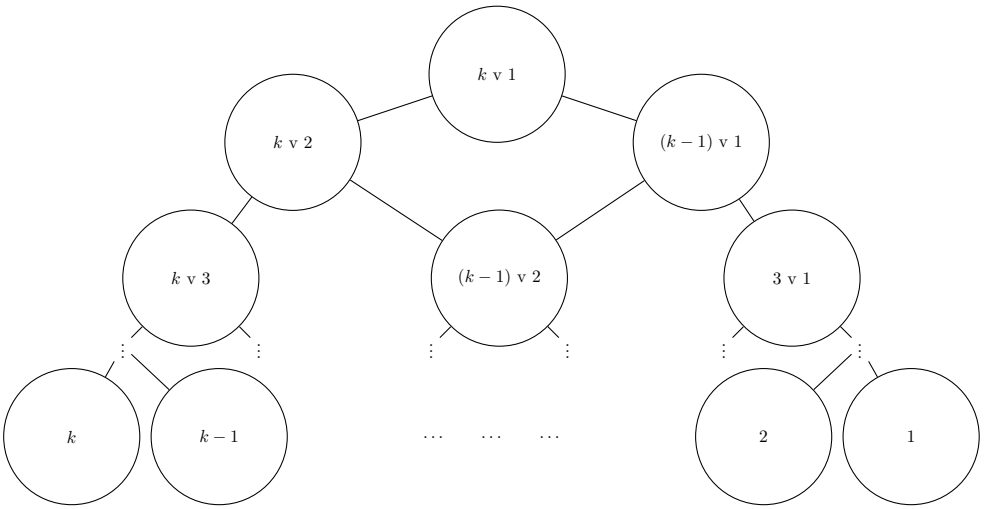


**Figure 4.4:** This figure illustrates the tree-like structure using a DAGSVM approach for a dataset of $k$ classes. At first data is compared between classes 1 and $k$ and one of the two classes is excluded depending on the result. Then data is classified between all additional non-excluded classes. When all pairwise tests are complete data is assigned to any of the $k$ classes. Note that the final assignment is not neccessary true but tells that the tested data is more similar to the assigned class than any of the other $k-1$ classes.

be seen in Figure 4.4.

## 4.3 Support vector machines for symmetric positive definite matrices

SVM can be generalized to account for data with a non-Euclidean structure. In this section we present data classification in the space of symmetric positive definite matrices. For certain types of data the classification accuracy can vastly improve if the model does not disregard underlying structure. The intuition behind this is the following: Imagine a set of data from two groups each distributed on a sphere of radius $r$. It is clear that there exists scenarios where the data is clearly separable. If distance is measured between the data by using an Euclidean distance, i.e. the length of a straight line connected between two points, and neglecting the fact that the data is on the sphere, one could still identify a clear separation between the two groups of data. However, one could also reformulate the implementation of the model, such that it accounts for the spherical structure. The spherical example is a very simple one but illustrative. In a similar way we implement a SVM with respect to the space of symmetric positive definite (SPD) matrices. This space is a *Riemannian manifold*. The theory of Riemannian manifolds is beyond the scope of this thesis but below we briefly outline the concepts needed to implement a model to account for the structure of the space of symmetric positive definite matrices.

A real symmetric positive definite (SPD) matrix $S$ of size $n \times n$ is a square matrix such

that for all $x \in \mathbb{R}^n$ we have $x^T S x > 0$. An extension to complex vectors is completely analogous but the transpose is replaced with the conjugate transpose. Consider the set of all $n \times n$ SPD matrices, which we denote by $\mathbb{S}$. The set $\mathbb{S}$ is an open convex cone which is a Riemannian manifold [12]. Briefly put, a manifold is a set which locally resembles Euclidean space. Moreover, a Riemannian manifold can loosely speaking be described as a manifold such that the inner product on the tangent spaces changes continuously. One example of a Riemannian manifold is the sphere in $\mathbb{R}^3$, i.e. a smooth two-dimensional surface in a three-dimensional space.

On a Riemannian manifold one can measure quantities such as distances. Thus, given two matrices $P$ and $Q$ on $\mathbb{S}$, $d(P, Q)$ is the distance between $P$ and $Q$. One can endow $\mathbb{S}$ with different metrics such as an *affine-invariant metric* (AIM), but in this thesis we consider the *log-Euclidean metric* (LEM). Both metrics are applicable but AIM usually comes with a high computational cost [14] compared to using a LEM. We begin by defining the logarithm of a matrix.

If $A \in \mathbb{S}$ then $A$ is diagonalizable as $A = U^T D U$ where $U$ is a matrix whose rows comprise an ON-basis and are the eigenvectors of $A$, and $D = \operatorname{diag}(\lambda_1, ..., \lambda_n)$ where $\lambda_i$ are the eigenvalues of $A$. That $A \in \mathbb{S}$ implies that $\lambda_i > 0$ for $i = 1, ..., n$. The logarithm of $A$ is by the spectral theorem given by

$$\log A = U^{-1} \operatorname{diag}(\log \lambda_1, \log \lambda_2, ..., \log \lambda_n) U \, .$$

If $P$ and $Q$ are elements on $\mathbb{S}$ then, in the log-Euclidean framework the distance between $P$ and $Q$ is

$$d(P, Q) \coloneqq \| \log P - \log Q \|_F \, , \tag{4.7}$$

where $\| \cdot \|_F$ denotes the Frobenius norm defined by

$$\|A\|_F \coloneqq \sqrt{\operatorname{Tr}(A A^\dagger)} \, , \tag{4.8}$$

and $A^\dagger$ is the conjugate transpose of the matrix $A$. Recall that $\operatorname{Tr}(A)$ is the sum of the diagonal elements of $A$.

With the log-Euclidean metric we can obtain a positive definite Gaussian kernel on $\mathbb{S}$ and equip the SVM model with this kernel to classify data on $\mathbb{S}$. A valid kernel on $\mathbb{S}$ is the *Log-Euclidean Gaussian kernel* defined as

$$\mathcal{K}(P, Q) \coloneqq \exp(-\gamma \| \log P - \log Q \|_F^2) \, , \tag{4.9}$$

where $\gamma > 0$, [16]. Henceforth SPD matrices are classified by solving problem (4.4) using the kernel specified in (4.9).

# Chapter 5

# Datasets and algorithms

In this chapter we describe the dataset that was used to test the proposed method for data classification. We also describe the algorithm used to solve the optimization problem that followed by using a SVM. In this thesis we focus on the MNIST dataset. This set is described in Section 5.1. Data is preprocessed using *deslanting* which is a type of "image rotation" to reduce within-class-variation. The preprocessing is described in 5.2. The procedure to classify images is described in Chapter 5.4.

## 5.1 MNIST

We test the proposed method for image classification using the MNIST dataset [8]. This dataset consists of images of handwritten digits for the digits $0, 1, ..., 9$. It consists of a training set with 60000 training images and a test set of 10000 images in total. All images are by the contributors size-normalized and centered. Each image have an equal size of $28 \times 28$ pixels and each pixel has a corresponding grey level value.

## 5.2 Preprocessing

The dataset was preprocessed by *deslanting* each image. The MNIST set comes with a large amount of images of the same digit but images within the same class are differently oriented or skewed. To reduce the amount of variation in each class we rotate each digit such that a least-squares regression line passing through the centroid of the image is rotated to a vertical position. This method has shown classification improvements for earlier work with the MNIST dataset [27]. If $I(X, Y)$ denotes the image with coordinates $X$ and $Y$ we define

$$\bar{X} := \frac{\sum_{X,Y} XI(X,Y)}{\sum_{X,Y} I(X,Y)},$$

$$\bar{Y} := \frac{\sum_{X,Y} YI(X,Y)}{\sum_{X,Y} I(X,Y)},$$

and finally

$$m := \frac{\sum_{X,Y} XY\, I(X,Y) - \bar{X}\bar{Y} \sum_{X,Y} I(X,Y)}{\sum_{X,Y} Y^2 I(X,Y) - \bar{Y}^2 \sum_{X,Y} I(X,Y)}\,.$$

We consider the regression line given by

$$X' = X + m(Y - \bar{Y})\,.$$

If we denote the floor and ceiling function respectively by $\lfloor\rfloor$ and $\lceil\rceil$, we deslant an image $I$ by the following mapping

$$I_{\text{deslant}}(X,Y) = (\lceil X' \rceil - X')I(\lfloor X' \rfloor, Y) + (X' - \lfloor X' \rfloor)I(\lceil X' \rceil, Y)\,.$$

Deslanting the image in the described way preserves the position of the centroid and skewing of the image is performed about the centroid.

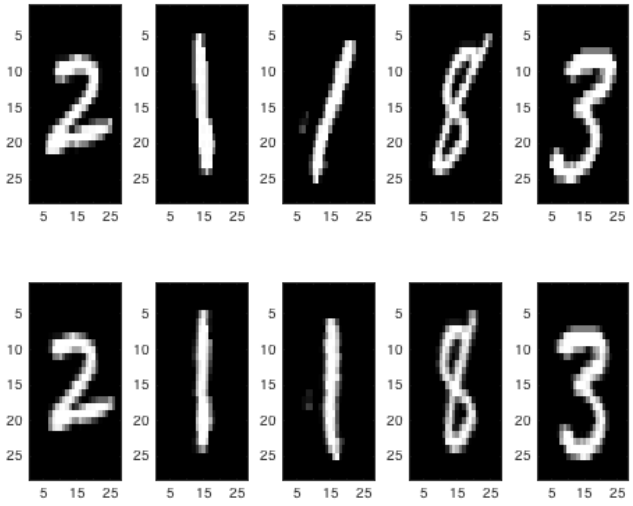

**Figure 5.1:** This figure illustrates the effect of deslanting the images. The top row corresponds to the original images while the bottom row consists of deslanted images. Note how some of the images are "more aligned" vertically after deslanting.

## 5.3 Repeated training on support vectors

Finally, we describe an additional method that is pursued as an attempt to improve classification accuracy. We refer to this method as *repeated training on support vectors* (RTSV). Assume we want to train a model using the classes $i$ and $j$ with $N$ data points per class. Ideally, we wish to train on as much significant data as possible, that is, train the model on non-redundant data. Data in a trained model that is non-redundant have

non-zero weights and therefore are significant components to the hyperplane. We also referred to data with non-zero dual weights as support vectors.

The idea behind this method is to mimic training on a larger dataset with at least $N$ data points by repeatedly training on smaller datasets. To find optimal weights for a binary classifier between the pair of classes $(i, j)$ we begin by training a model with $n$ data points per class where we choose $n < N$. Of course one can choose $n = N$ however we choose $n < N$ to reduce the computational cost. If $s_{i1}$ and $s_{j1}$ denotes the number of support vectors obtained from the trained model in the initial run, we store these for later use. We repeat training on classes $i$ and $j$ using $n$ new data points from each class, assume we obtain $s_{i2}$ and $s_{j2}$ support vectors from class $i$ and $j$ respectively. This procedure is repeated until we have found $N$ support vectors from each class, i.e. assuming it took $k$ runs to find $N$ unique support vectors, we have $N = \sum_k s_{ik} = \sum_k s_{jk}$ from each class. Each support vector must be unique since we do not benefit from training on duplicates of data.

The final binary classifier is constructed by training on the $2N$ previous support vectors and classify data in the usual manner to build a final model.

## 5.4 Construction of correlation matrices

In this chapter we describe how the classification of the MNIST dataset is performed.

Recall that all images are preprocessed using deslanting. To simplify notation, we denote a deslanted image by $I$. Since the models using shearlet or Gabor coefficients are completely identical, except from the fact that one model uses shearlet coefficients, and the other one Gabor coefficients, we describe the process from the shearlet point of view. However again note that the only difference is the corresponding transform coefficients.

Using the deslanted images we proceed by shearlet transforming each image $I$ using (2.3.4) introduced in Chapter 2.3. This results in $2^{j+2}$ matrices of shearlet coefficients, which we denote by $\psi_{j,k}$. Since the images were small, $28 \times 28$ pixels, we transformed for $j = 1$. Using the shearlet coefficients $\psi_{j,k}$ and the image $I$ with coordinates $X$ and $Y$, we form the feature matrix

$$
\begin{aligned}
\boldsymbol{f} &= \\
&= [X\ Y\ I(X,Y)\ \psi_{1,-2^j}(X,Y)\ \dots\ \psi_{1,2^j}(X,Y)\ \dots\ \psi_{1,-2^j+1}(X,Y)] \\
&= \begin{pmatrix}
x_1 & y_1 & I(x_1,y_1) & \psi_{1,-2^j}(x_1,y_1) & \dots & \psi_{1,2^j}(x_1,y_1) & \dots & \psi_{1,-2^j+1}(x_1,y_1) \\
x_2 & y_1 & I(x_2,y_1) & \psi_{1,-2^j}(x_2,y_1) & \dots & \psi_{1,2^j}(x_2,y_1) & \dots & \psi_{1,-2^j+1}(x_2,y_1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{28} & y_1 & I(x_{28},y_1) & \psi_{1,-2^j}(x_{28},y_1) & \dots & \psi_{1,2^j}(x_{28},y_1) & \dots & \psi_{1,-2^j+1}(x_{28},y_1) \\
x_1 & y_2 & I(x_1,y_2) & \psi_{1,-2^j}(x_1,y_2) & \dots & \psi_{1,2^j}(x_1,y_2) & \dots & \psi_{1,-2^j+1}(x_1,y_2) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{28} & y_{28} & I(x_{28},y_{28}) & \psi_{1,-2^j}(x_{28},y_{28}) & \dots & \psi_{1,2^j}(x_{28},y_{28}) & \dots & \psi_{1,-2^j+1}(x_{28},y_{28})
\end{pmatrix}.
\end{aligned}
$$

where $I$ and each matrix $\psi_{j,k}$ forms a vector by concatenating each row after one another. For example consider the identity matrix of size $3 \times 3$. Concatenating each row forms the vector $(1,0,0,0,1,0,0,0,1)$. Moreover, since each image is of size $28 \times 28$, $\boldsymbol{f}$ is of size $28^2 \times (3 + 2^{j+2})$. Using $\boldsymbol{f}$ we calculate the correlation between each column in $\boldsymbol{f}$, this gives a correlation matrix which we denote by $F$. The matrix $F$ is of size $(3 + 2^{j+2}) \times (3 + 2^{j+2})$ (note that $F$ is independent of the amount of pixels in $I$). The matrix element $F(m,n)$ corresponds to the correlation between feature $m$ and $n$.

For each image $I$ we obtain a corresponding matrix $F$. Since $F$ is a correlation matrix $F$ is symmetric positive definite (SPD) since it is a correlation matrix. Moreover we know that the set of SPD matrices of size $n \times n$ constitutes a Riemannian manifold. We can therefore measure the distance between each matrix $F$ using the metric defined in (4.7). Hence using the matrices $F$ we construct a soft-margin SVM. The SVM model consists of solving the optimization problem (4.4) using the kernel defined in (4.9). A new image is classified by constructing its corresponding matrix $F$ and the mapping defined in (4.6).

## 5.5 Algorithms

All coding were done in MATLAB. The shearlet transform was implemented as described in [22]. The shearlet transform requires multidimensional Fourier transforms and we used `fft2` and `ifft2` available through MATLAB. To solve the optimization problem (4.4) introduced in Chapter 4 we used `fmincon` with solver `SQP` (sequential quadratic programming).

## 5.6 Choice of parameters

In our model we have two parameters $\lambda$ and $\gamma$. The parameter $\lambda$ puts an upper boundary on the dual weights $c_i$ as shown in (4.2). The parameter $\gamma$ is related to the gaussian kernel which we previously defined in (4.9). Here we describe how these two were determined.

The parameter lambda was determined by repeatedly classify randomly picked data as we change $\lambda$. Note that the maximum value of the weights $c_i$ is $\frac{1}{2n\lambda}$ where $n$ is the total training size. In our simulations we used $\lambda = \frac{1}{100n}$.

To determine the parameter $\gamma$ we looked at the function $f(x) = e^{-\gamma x^2}$ since we used the kernel (4.9). First, we note that $0 \leq f \leq 1$ for all $x \in \mathbb{R}$. The idea is that we want to map the distance between the matrices as spread out between 0 and 1 as possible. This implies that we want to choose $\gamma$ such that the majority of the arguments $x$ to $f$ is as near the steepest part of $f$ as possible. The steepest part of $f$ is at its inflection point. Thus we want to choose $\gamma$ such that the argument to $f$ maps as near the inflection point as possible. Denote the inflection point by $x_0$. Then $x_0$ is the solution to $f''(x_0) = 0$, i.e.

$$f''(x) = 0 \Leftrightarrow 2\gamma e^{-\gamma x^2}(2\gamma x^2 - 1) = 0 \Rightarrow x_0 = \frac{1}{\sqrt{2\gamma^2}}\,.$$

Since we wish to map the majority of the data to a point near the inflection point, we let $\mu$ denote the mean distance (using the log-Euclidean metric) between the data points in a binary classifier, and choose $\gamma = \frac{1}{2\mu^2}$.

# Chapter 6

# Classification performance

In this section we present the classification results using the shearlet and Gabor based SVM. We begin by presenting the classification performance between the 90 binary classifiers that are derived using the MNIST dataset. We proceed with the classification results using a DAGSVM.

## 6.1 Performance of binary classifiers

The MNIST dataset consists of 10 classes that corresponds to the integers $0, 1, ..., 9$. Generally speaking, for a dataset of $k$ classes we can construct $k(k-1)$ binary classifiers. However comparing $(i, j)$ is equivalent to $(j, i)$ since the only difference is the assignment of label $+1$ or $-1$, and therefore we have $k(k-1)/2$ unique classifiers for $k$ classes of data. To measure the performance of each binary classifier $(i, j)$ we do the following:

1. Pick two classes $i$ and $j$.
2. Randomly pick $N = 400$ data points without replacement from each class. This results in $2N$ total points for training.
3. Find the optimal weights $c_i$ where $i = 1, ..., 2N$ and calculate the bias to construct a model for pair $(i, j)$ using the $2N$ data points.
4. Randomly pick 500 **new** data points for testing from class $i$ and class $j$ without replacement. This gives a total points for testing equal to 1000.
5. Classify the 1000 points using the binary classifier constructed in step 1-3, and count the amount of incorrect classifications.
6. Repeat the steps above four times for the same pair $(i, j)$ and calculate the mean incorrect classifications, denote this number by $\mu(i, j)$.

We repeat the process above for each pair of classes $(i, j)$ except for $i = j$ using both the shearlet and Gabor based model. The reasoning behind randomly drawing samples from

the full sets is to generalize the model and measure the performance by continuously training the model, and exposing it to new but similar data.

The results obtained are shown in Table 6.1 and 6.3. To compare the two tables we color the values blue or red depending on the performance between the two. For two models A and B we say that model $A$ was more accurate than model $B$ if $A$ was at least 0.4% more accurate than $B$ for a specific pair of classes $(i, j)$. If any value is coloured blue, it means that the result from that specific binary classifier was better than its contender (shearlet or Gabor). For example if $\mu(i, j)$ is blue in Table 6.1, then for the same pair $(i, j)$ the value $\mu(i, j)$ is red in Table 6.3.

Recall that the only difference between classifying $(i, j)$ or $(j, i)$ is the label assignment. Nevertheless we present $k(k-1)$ results instead of $k(k-1)/2$ results in each table. The reason behind this is to show that the results are independent of the assignments of labels since any difference between $(i, j)$ and $(j, i)$ is due to small within-class-variations in the dataset.

**Table 6.1:** The mean error rate between the binary classes using deslanting and the shearlet feature extraction. The total mean error in this table is 1.93%, median 1.33% and standard deviation 1.53%. We see that the majority of the values are smaller compared to the results obtained using the Gabor model, which is indicated by the color blue.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0.30% | 1.25% | 0.88% | 0.60% | 1.08% | 2.13% | 0.80% | 1.52% | 1.18% |
| 1 | 0.43% | | 0.65% | **0.23%** | 0.83% | 0.28% | 0.63% | 0.90% | 0.68% | 0.73% |
| 2 | 1.43% | 0.45% | | 4.45% | 1.43% | 4.35% | 4.20% | 1.98% | 3.28% | 2.00% |
| 3 | 0.50% | **0.20%** | 4.78% | | 0.63% | **6.70%** | 1.33% | 1.33% | 4.40% | 2.18% |
| 4 | 0.55% | 1.05% | 1.13% | 0.53% | | 0.56% | 1.28% | 3.33% | 1.68% | 4.03% |
| 5 | 1.05% | 0.40% | 4.70% | **6.90%** | 0.65% | | 1.48% | 1.10% | 4.08% | 2.93% |
| 6 | 1.83% | 0.65% | 4.13% | 1.23% | 1.35% | 1.34% | | 0.48% | 2.15% | 1.05% |
| 7 | 0.53% | 0.63% | 1.68% | 0.85% | 2.73% | 1.15% | 0.90% | | 1.40% | 4.65% |
| 8 | 1.60% | 0.48% | 3.28% | 4.33% | 1.28% | 3.78% | 2.40% | 1.45% | | 3.83% |
| 9 | 1.50% | 0.73% | 2.33% | 2.30% | 3.40% | 2.58% | 1.13% | 4.23% | 3.03% | |

**Table 6.2:** Results from repeated training on support vectors using shearlets. The testing procedure is performed as described in Section 6.1.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0.08% | 0.88% | 0.30% | 0.20% | 0.35% | 1.05% | 0.03% | 1.03% | 0.55% |
| 1 | | | 0.53% | 0.05% | 0.50% | 0.05% | 0.38% | 0.48% | 0.15% | 0.68% |
| 2 | | | | 3.60% | 0.55% | 3.45% | 3.48% | 1.18% | 2.23% | 1.85% |
| 3 | | | | | 0.25% | **5.60%** | 0.85% | 0.43% | 3.55% | 1.55% |
| 4 | | | | | | 0.35% | 0.60% | 2.20% | 0.83% | 2.58% |
| 5 | | | | | | | 1.45% | 0.50% | 2.58% | 1.68% |
| 6 | | | | | | | | 0.65% | 1.70% | 0.90% |
| 7 | | | | | | | | | 0.93% | 3.68% |
| 8 | | | | | | | | | | 2.75% |
| 9 | | | | | | | | | | |

**Table 6.3:** The mean error rate between the binary classes using the Gabor feature extraction. Here the testing and training is identical to the results shown in 6.1 except that we used the Gabor wavelet instead of the shearlet. The total mean error in this table is 2.45%, median 1.98% and a standard deviation of 1.97%. We see that the majority of the values are higher (larger error) compared to the results obtained using the shearlet model, which is indicated by the color red.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0.40% | 4.45% | 1.43% | 1.10% | 2.13% | 2.65% | 0.55% | 0.98% | 1.83% |
| 1 | 0.43% | | 0.60% | **0.30%** | 0.63% | **0.30%** | 0.70% | 0.58% | 0.40% | 0.53% |
| 2 | 4.08% | 0.63% | | 5.80% | 1.89% | 8.10% | 6.98% | 2.15% | 3.23% | 3.70% |
| 3 | 1.58% | **0.20%** | 5.35% | | 0.98% | **8.15%** | 3.33% | 1.30% | 2.63% | 1.98% |
| 4 | 1.48% | 0.73% | 2.85% | 0.53% | | 1.25% | 4.15% | 3.00% | 1.45% | 3.80% |
| 5 | 2.15% | **0.35%** | 6.98% | **7.90%** | 0.88% | | 3.45% | 2.08% | 1.33% | 3.35% |
| 6 | 2.83% | 0.43% | 7.20% | 2.83% | 4.58% | 3.05% | | 1.70% | 4.18% | 3.90% |
| 7 | 0.58% | 0.80% | 2.10% | 1.25% | 2.80% | 1.83% | 1.85% | | 0.80% | 3.03% |
| 8 | 1.13% | 0.48% | 3.83% | 2.55% | 1.55% | 1.53% | 4.70% | 0.73% | | 2.98% |
| 9 | 1.90% | 0.80% | 3.70% | 2.43% | 3.98% | 2.83% | 3.40% | 3.10% | 3.18% | |

The results shown in the Tables 6.1 and 6.3 indicate that in general the shearlet based binary classifiers performs better compared to the Gabor based ones. The shearlet model performs slightly worse than Gabor for class 8, they are almost equal in terms of

classifying class 1, but for the other classes the shearlet based model performs better in every single case. This is also indicated by the corresponding mean error of all binary classifiers. The mean error of all shearlet based classifiers was 1.93%, while the mean error for the Gabor based classifiers was 2.45%.

## 6.2 Performance using a DAGSVM

In this section we represent the classification performance of the MNIST dataset using a MSVM based on the shearlet and Gabor transform. We construct the MSVM using a DAGSVM which was described in Section 4.2. In a classification model, the training size can have major effects on classification accuracy. For this we used three different sizes, namely $N = 300, 500, 700$ per class. We test the entire test set that is obtainable from MNIST. The MSVM is extremly slow compared to the binary classifiers. This is mainly due to the training sizes but also since each testing point has to be classified 10 times before it gets a final assignment. For example classifying all the data (a total of 10000 points) corresponds to a total of $10^5$ tests. When the total training size is larger than $10^3$ it takes several hours to find the minimum that corresponds to the optimal hyperplane. Therefore we were not able to pursue training on the full training set as desired. However when making a linear increase of total training size we note an almost linear increase in classification accuracy. In general the shearlet based model performed better than the Gabor based model for both the binary- and multiclass models. The difference between shearlet and Gabor for some binary classifiers were almost zero but shearlet outperformed the Gabor ones for the majority of the classifiers. The most significant difference was that shearlets performed much better at classifying digit 2 compared to Gabor wavelets. The shearlet based binary classifiers had a harder time classifying digit 8 compared to Gabors, this is also visible in the MSVM results shown in Tables 6.4 and 6.5.

**Table 6.4:** The results from classifying all images in the MNIST test dataset using a shearlet based DAGSVM. Here $\mathcal{E}$ denotes the number of incorrect classifications, and $\mu(\mathcal{E})$ is the mean of $\mathcal{E}$ (in percent) averaged over all 10 classes. The first row shows the order of the corresponding class (MNIST digit). The second row is the size of the full test set, and the following rows correspond to the number of incorrect classifications, for different training sizes $N$. Note that the number $N$ is per class, thus the full training size for each binary classifier is $2N$. The rows marked by RTSV is the results from repeatedly training on support vectors.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\mu(\mathcal{E})$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Size of class | 980 | 1135 | 1032 | 1010 | 982 | 892 | 958 | 1028 | 974 | 1009 | |
| $\mathcal{E}$ (N=300) | 61 | 25 | 159 | 115 | 84 | 108 | 58 | 76 | 132 | 137 | 9.70 |
| $\mathcal{E}$ (N=300)(RTSV) | 20 | 11 | 105 | 91 | 42 | 75 | 44 | 65 | 98 | 91 | 6.49 |
| $\mathcal{E}$ (N=500) | 39 | 31 | 150 | 104 | 59 | 90 | 59 | 70 | 117 | 102 | 8.27 |
| $\mathcal{E}$ (N=500)(RTSV) | 21 | 11 | 111 | 79 | 46 | 80 | 46 | 49 | 101 | 76 | 6.28 |
| $\mathcal{E}$ (N=700) | 34 | 19 | 125 | 102 | 55 | 83 | 56 | 72 | 115 | 96 | 7.64 |
| $\mathcal{E}$ (N=700)(RTSV) | - | - | - | - | - | - | - | - | - | - | - |

**Table 6.5:** The results from classifying all images in the MNIST test dataset using a Gabor based DAGSVM. Here $\mathcal{E}$ denotes the number of incorrect classifications, and $\mu(\mathcal{E})$ is the mean of $\mathcal{E}$ (in percent) averaged over all 10 classes. The first row shows the order of the corresponding class (MNIST digit). The second row is the size of the full test set, and the following rows correspond to the number of incorrect classifications, for different training sizes $N$. Note that the number $N$ is per class, thus the full training size for each binary classifier is $2N$.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\mu(\mathcal{E})$ [%] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Size of class | 980 | 1135 | 1032 | 1010 | 982 | 892 | 958 | 1028 | 974 | 1009 | |
| $\mathcal{E}$ (N=300) | 76 | 20 | 254 | 131 | 100 | 153 | 123 | 110 | 86 | 131 | 11.98 |
| $\mathcal{E}$ (N=300)(RTSV) | 53 | 10 | 218 | 130 | 67 | 131 | 107 | 50 | 71 | 99 | 9.49 |
| $\mathcal{E}$ (N=500) | 56 | 17 | 232 | 131 | 78 | 133 | 118 | 75 | 104 | 130 | 10.87 |
| $\mathcal{E}$ (N=500)(RTSV) | 49 | 8 | 198 | 120 | 62 | 125 | 108 | 58 | 67 | 107 | - |
| $\mathcal{E}$ (N=700) | 67 | 18 | 224 | 114 | 61 | 117 | 93 | 73 | 78 | 108 | 9.63 |
| $\mathcal{E}$ (N=700)(RTSV) | - | - | - | - | - | - | - | - | - | - | - |

It is interesting to note that classification accuracy for some of the classes appears to remain almost constant for different values of $N$, while some are slightly improving. Classification of class 0, 4 and 9 however have a more rapid change compared to the other classes which indicates that the classification accuracy could be improved more by on more samples. Generally for all classes, the classification accuracy is improving by

increases the training size, thus it is plausible that even greater results are obtained if $N$ is increased more.

# Chapter 7

# Conclusion

Our simulations indicate that the shearlet transform performs better for image classification of the MNIST dataset compared to the Gabor transform. We use the notion of 'indicate' in the sense that larger training sets and further improved feature extraction are needed to actually say that shearlets are better. All MSVM experiments show that shearlet performs slightly better than the Gabor since the relative difference between the results were approximately 20%. As for the binary cases the shearlet also performs better for the majority of the cases, however the Gabor is slightly better at classifying digit 8. Moreover the error for some binary classifiers were almost equal indicating that the two models classified those classes equally well.

Compared to other algorithms applied to the MNIST dataset our results are not particularly outstanding. This is most notably due to the not being able to apply the models for the entire MNIST dataset. However the goal of this thesis was to simply compare shearlets with wavelets, and not to construct a model that outperforms other current state-of-the-art algorithms. If the goal of this thesis was to construct an algorithm to give outstanding classification result it had been better to use already available SVM-packages that are also optimized for data classification using support vector machines. However from a learning perspective of the author, not using available packages was successful.

## 7.1 Future work and choice of algorithm

As with any algorithm, there are many ways to construct and modify an algorithm. Therefore, in this section we keep a discussion regarding the advantages or disadvantages of the proposed model and point out some interesting ideas for future work.

It would also be interesting to construct a model that is not based on correlation matrices, but instead, a model that sees the actual transform coefficients. Since perhaps the correlation matrix is an abstract way of representing an image and therefore using the coefficients themselves as input to the model is perhaps a better approach. If not correlation matrices pursue covariance matrices.

The shearlet transform is capable of transforming an image for different scales $j$.

The amount of scales is limited to the size of the image in process. In the MNIST set, all images were of size $28 \times 28$ which could be considered typically small. This implies that we can only consider the smallest scales of $j$ such as $j = 0$ and $j = 1$ for any useful information. If the images were larger we could obtain more precise information regarding edges in the images from $j = 2, 3$ and so forth.

The model in this thesis is independent of the size of the image, since we use the transformation coefficients to construct correlation matrices. The size of those matrices depend only on the amount of features. Adding additional features does neither increase the computational cost by any noticable amount. The only time-consuming part in the algorithm is finding a solution to the optimization problem. The optimization is based on `fmincon` and solves a high dimensional problem. If we have $N$ data points for training in a binary classifier, `fmincon` tries to find a minimum for a function of $N$ variables. The MNIST dataset makes it possible to train on several thousands of data points per class but this is practically not doable due to the proposed implementation being inefficient for large datasets.

Using the correlation matrices it is also possible to add additional information on the diagonal of the correlation matrix, for example such as mean or variances, additional color informaion (if it is available) etc. There are also several techniques for preprocessing of data. Here we only used deslanting and it improved the classification accuracy by a few percent. There might be additional methods that could improve the classification accuracy of the proposed method.

Another interesting idea is to test other metrics and kernels. We did only consider the log-Euclidean metric using a Gaussian RBF kernel. However there are several different metrics and kernels that could also be applied to this problem. For example the affine-invariant metric and stein metric, and kernels such as polynomial-based kernels.

Since current state-of-the-art algorithms, i.e. algorithms based on neural networks, currently classify data to a very high accuracy one might ponder upon why anyone would use shearlets for data classification. In fact these models are well suited for image classification but currently not applicable for video classification. There are previous work of video classification using a shearlet approach and Riemannian manifolds in [12] where the author reached very good classification results. Using the shearlet transform and Riemannian manifolds one can define velocities on the manifold i.e. rate of change in a video which obviously can be very important information in video classification. Also since a video is a sequence of image, one can define a path that is defined by the location of each image on that manifold. Moreover one can turn to the three-dimensional shearlet transform which could also be applicable for data classification.

The next step for shearlets and image classification is to investigate how the shearlet framework can be combined with current state-of-the-art algorithms to possibly obtain even greater results.

# Bibliography


[1] A. Graps, An introduction to wavelets, IEEE Computational Science and Engineering 2 (2) (1995) 50–61.

[2] G. Easley, D. Labate, W.-Q. Lim, Sparse directional image representations using the discrete shearlet transform, Applied and Computational Harmonic Analysis 25 (1) (2008) 25 – 46.

[3] K. Guo, G. Kutyniok, D. Labate, Sparse multidimensional representations using anisotropic dilation and shear operators, Mod. Methods Math., Nashboro Press, Brentwood, TN, 2006.

[4] G. R. Easley, D. Labate, W.-Q. Lim, Optimally sparse image representations using shearlets (2006).

[5] S. Thayammal, D. Selvathi, Edge preserved multispectral image compression using extended shearlet transform, The Computer Journal (2016, June 17).

[6] S. Zhou, J. Shi, J. Zhu, Y. Cai, R. Wang, Shearlet-based texture feature extraction for classification of breast tumor in ultrasound image, Biomedical Signal Processing and Control 8 (6) (2013) 688 – 696.

[7] Y. Qu, X. Mu, L. Gao, Z. Liu, Facial Expression Recognition Based on Shearlet Transform, 2012th Edition, Vol. 159, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, Ch. 1, p. 559–565.

[8] Y. LeCun, C. Cortes, MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist/ (2010).

[9] S. Yi, D. Labate, G. R. Easley, H. Krim, A shearlet approach to edge analysis and detection, IEEE Transactions on Image Processing 18 (5) (2009) 929–941.

[10] W. Jiang, K.-M. Lam, T.-Z. Shen, Efficient edge detection using simplified Gabor wavelets, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (4) (2009) 1036–1047.

[11] X. Wang, X. Ding, C. Liu, Gabor filters-based feature extraction for character recognition, Pattern Recognition 38 (3) (2005) 369–379.

[12] Y. Yun, C. U. of Technology, S. of Electrical Engineering, C. tekniska högskola, S. Institutionen för signaler och system, D. of Signals, S. P. Systems, Riemannian manifold-based modeling and classification methods for video activities with applications to assisted living and smart home (2016).

[13] H. Q. Minh, V. Murino, S. O. service), S. (e-book collection), Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization, Springer International Publishing, Cham, 2016.

[14] Z. Huang, R. Wang, S. Shan, X. Li, X. Chen, Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, p. 720–729.

[15] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (1) (2008) 1–37.

[16] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, M. Harandi, Kernel methods on Riemannian Manifolds with Gaussian RBF kernels, IEEE transactions on pattern analysis and machine intelligence 37 (12) (2015;2014;) 2464–2477.

[17] J. Bergh, F. Ekstedt, M. Lindberg, Wavelets, Studentlitteratur, Lund, 1999.

[18] C.-L. Liu, A tutorial of the wavelet transform (2010).

[19] D. Barina, Gabor wavelets in image processing, CoRR abs/1602.03308.

[20] R. Mehrotra, K. Namuduri, N. Ranganathan, Gabor filter-based edge detection, Pattern Recognition 25 (12) (1992) 1479 – 1494.

[21] D. Labate, G. Kutyniok, Shearlets, 2012th Edition, Springer Verlag, DE, 2012.

[22] S. Häuser, G. Steidl, Fast finite shearlet transform (2012).

[23] W. Commons, Frequency tiling of a classical shearlet system, file: `Classsheartiling.svg` (2013).
URL `https://commons.wikimedia.org/wiki/File:Classsheartiling.svg`

[24] G. Kutyniok, M. Shahram, D. L. Donoho, Development of a digital shearlet transform based on pseudo-polar fft (2009).

[25] I. Steinwart, A. Christmann, S. (e-book collection), Support vector machines, 1st Edition, Springer, New York, 2008.

[26] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2) (2002) 415–425.

[27] L.-N. Teow, K.-F. Loe, Robust vision-based features and classification schemes for off-line handwritten digit recognition, Pattern Recognition 35 (11) (2002) 2355–2364.

[28] H. König, S. O. service), S. A. (e-book collection), Eigenvalue Distribution of Compact Operators, Vol. 16, Birkhäuser Basel, Basel, 1986.