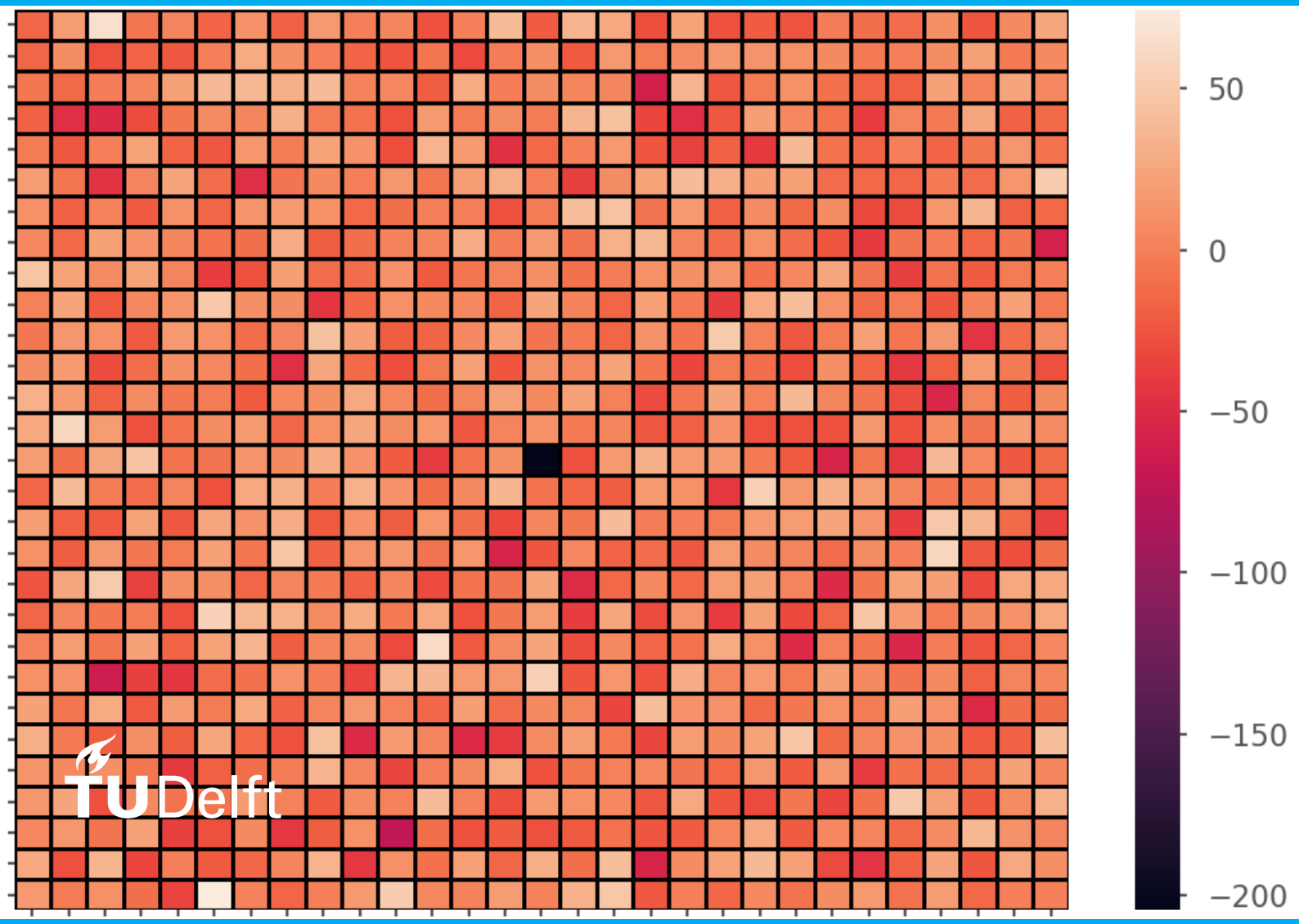
On the Influence of Whitening Transformations on Hyperspectral Data

G.K. van der Wal



On the Influence of Whitening Transformations on Hyperspectral Data

by

G.K. van der Wal

to obtain the degree of Master of Science in Applied Mathematics
at the Delft University of Technology,
to be defended on the 26th of September, 2023 at 13:30.

Student number:	4602501	
Project duration:	September 2022 – September 2023	
Thesis committee:	Prof. dr. ir. M. B. van Gijzen,	TU Delft, thesis advisor
	Dr. J. Söhl,	TU Delft, supervisor
	Dr. R. G. Satink,	Stage Gate 11 B.V., supervisor

This thesis is confidential and will be made public two years after the defence date.

An electronic version of this thesis will be made available at `http://repository.tudelft.nl/`.

TU Delft

Stage Gate 11

Preface

This is the thesis “On the Influence of Whitening Transformations on Hyperspectral Data", where synthetic data is created to research the influence of whitening on a target spectrum and its signal-to-noise ratio. The synthetic data is further used to create high contrast scenario's, to see the influence of this on a target spectrum, and how multi-area whitening can help with the classification of illegal substances.

This thesis was written as part of a graduation project at Stage Gate 11 B.V., in order to obtain the degree of Master of Science in Applied Mathematics at Delft University of Technology. It is also meant for the employees of Stage Gate 11 B.V., to help them with their classification algorithm. This thesis was written in the period from September 2022 to September 2023 at Stage Gate 11 B.V. in Breukelen, under the supervision of Rob Satink, and in Delft, under the supervision of Jakob Söhl.

I would like to thank both Rob Satink and Jakob Söhl for their supervision, valuable feedback and support. Jakob, your expertise in mathematics has helped greatly with making the background in this thesis mathematically sound, due to your preciseness. I could not have done this on my own. I would also like to express me gratitude for supporting me through the writing process, which has not been easy for me. Rob, I am grateful for all the meetings we have had where you really got involved in the project. Your willingness to share your knowledge in spectroscopy and engage in discussions have truly shaped this thesis. Thank you for your guidance and support.

I would also like to extend my appreciation to the entire staff of Stage Gate 11 B.V., who where all willing to help where possible and open for questions and discussions. Last but not least, to my friends and family, thank you for your support and encouragement throughout this process.

G.K. van der Wal
Breukelen, September 18, 2023

Abstract

The whitening transformation transforms a random matrix into a whitened matrix with expectation 0 and covariance matrix I. By removing the first and second order statistical structures, higher order structures can be looked at for better classification. This is why Stage Gate 11 B.V. has employed whitening in the preprocessing of their hyperspectral data. The aim of this work is to gain insight into the whitening transformation and how it influences hyperspectral data.

To gain this insight, synthetic data was created and used to make synthetic scans. The signal-to-noise ratio of a target spectrum was calculated, and Monte Carlo simulations were used to reveal hidden patterns in the data. In case of a high contrast scenario, multi-area whitening was employed and the cosine similarity between the target spectrum and its signature was determined.

It was observed that the shape and intensity of the whitened target spectrum differs, depending on if pixels were used as observations or wavelengths. However, both are subject to the 'bleeding' effect. Further, it was found that if the number of pixels in the scan is greater than the number of spectral bands (548), then the signal-to-noise ratio becomes better as the number of whitened pixels in the scan increases. In case of a high contrast scenario, multi-area whitening guarantees the uniformity of the spectra, resulting in a higher cosine similarity between the target spectrum and its signature. But as multi-area whitening uses a smaller number of pixels in the scan, it cannot be concluded if multi-area whitening is better than global whitening, as it is not known how the increase in cosine similarity and the decrease in signal-to-noise ratio relate to the classification process. Finally, it is concluded that when working with real and unknown data, using pixels as observations is much more feasible.

List of Variables

Variable	Description	Note	Formula
Im	Input image of a scan		
μ_i	Mean over all wavelengths for pixel i		$\frac{1}{m}\sum_{j=1}^{m} Im_{i,j}$
X	Row-wise centered data set	Row average = 0	$X_{i,k} = Im_{i,k} - \mu_i$
Σ	Covariance matrix of X	Symmetric	$\Sigma = \mathbb{E}\left[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T\right]$
$\hat{\Sigma}$	Empirical covariance matrix of X	Symmetric	$\hat{\Sigma} = \frac{1}{m-1} XX^T$
X_i	Spectrum of pixel i		
W	Whitening matrix		$W^T W = \Sigma^{-1}$
Y	Whitened matrix	$\mathbb{E}[Y] = 0,\ \mathrm{Cov}(Y) = I$	$Y = WX$
V	Matrix containing the eigenvectors of Σ as its columns	Orthogonal	$\Sigma = VDV^T$
D	Matrix containing the corresponding eigenvalues of Σ	Diagonal	$\Sigma = VDV^T$
U_1	Rotation matrix	Orthogonal	$W = U_1\Sigma^{-1/2}$
U_2	Rotation matrix	Orthogonal	$W = U_2 P^{-1/2} S^{-1/2}$
S	Diagonal covariance matrix	Diagonal	$S = \mathrm{diag}(\Sigma)$
P	Correlation matrix	Symmetric	$\Sigma = S^{1/2} P S^{1/2}$
G	Matrix containing the eigenvectors of P as its columns	Orthogonal	$P = G\Theta G^T$
Θ	Matrix containing the corresponding eigenvalues of P	Diagonal	$P = G\Theta G^T$
R	Relation matrix between U_1 and U_2, $U_1 = U_2 R$	Orthogonal	$R = P^{-1/2} S^{-1/2} \Sigma^{1/2}$
Φ	Cross-covariance between Y and X		$\Phi = U_1\Sigma^{1/2}$
Ψ	Cross-correlation between Y and X		$\Psi = U_2 P^{1/2}$
C	Column-wise centered data set	Column average = 0	$C = X^T$
p_i	Principal component i of Σ	Orthonormal vectors	$V = (p_1, \ldots, p_n)$
Z	Row-wise centered data set, set of signatures	Row average = 0	
H	Matrix used in CCA		$H = \Sigma_X^{-1}\Sigma_{XZ}\Sigma_Z^{-1}\Sigma_{ZX}$
α_i	Canonical direction i of X, eigenvector of H	Orthonormal vectors	
η_i	i^{th} largest eigenvalue of H		$H\alpha_i = \eta_i \alpha_i$
β_i	Canonical direction i of Z	Orthonormal vectors	$\beta_i = r \cdot \Sigma_Z^{-1}\Sigma_{ZX}\alpha_i$
L	Matrix that is part of the Cholesky decomposition of Σ	Lower triangular	$\Sigma^{-1} = LL^T$
$\tilde{\Sigma}$	Shrinkage estimator of Σ	Symmetric	$\tilde{\Sigma} = \delta T + (1-\delta)\hat{\Sigma}$
T	Structured estimator for Σ	Symmetric	
δ	Shrinkage intensity		$\delta \in [0,1]$
δ^*	Optimal shrinkage intensity		

Contents

Preface iii

Abstract v

List of Variables vii

1 Introduction 1

2 Background 3
2.1 Hyperspectral Imaging . . . 3
2.2 The Covariance Matrix . . . 4
2.3 Whitening Tranformations . . . 5
2.3.1 Principal Component Analysis (PCA) . . . 7
2.3.2 Standardized PCA . . . 10
2.3.3 Zero-phase Component Analysis (ZCA) . . . 10
2.3.4 Standardized ZCA . . . 11
2.3.5 Canonical Correlation Analysis (CCA) . . . 12
2.3.6 Cholesky Whitening . . . 14
2.3.7 Characteristics of Whitening Transformations . . . 15

3 Methodology 17
3.1 Synthetic Data . . . 17
3.2 Whitening. . . . 20
3.2.1 Empirical Whitening Process. . . . 20
3.2.2 Used Whitening Transformations . . . 20
3.2.2.1 PCA and Standardized PCA Whitening . . . 20
3.2.2.2 ZCA and Standardized ZCA Whitening . . . 21
3.2.2.3 CCA Whitening . . . 21
3.2.2.4 Cholesky Whitening. . . . 21
3.3 Signal-to-Noise Ratio . . . 22
3.4 The Influence of the Number of Pixels in the Scene . . . 23
3.4.1 Monte Carlo Simulations. . . . 24
3.5 Multi-area whitening . . . 24
3.5.1 Creating the Leather Spectrum. . . . 24
3.5.2 The Influence of Shifting the Spectrum on the Scenario . . . 26
3.5.3 Spectral Angle and Cosine Similarity . . . 27
3.5.4 Monte Carlo Simulations. . . . 28

4 Results and Discussion 31
4.1 Without Shot Noise . . . 31
4.2 Signal-to-Noise Ratio . . . 34
4.2.1 Cutting of the Spectrum . . . 36
4.3 Multi-Area Whitening . . . 36

5 Conclusions 39

6 Recommendations 41

Bibliography 43

A The Empirical Covariance Matrix: Additional Proofs 45

B Figures 47

C Code 53
C.1 whitening_math_functions . . . 53
C.2 whitening_methods_functions . . . 56
C.3 whitening_SNR_functions . . . 59
C.4 Whitening_synthetic_data . . . 61
C.5 SNR_Monte_Carlo . . . 68
C.6 Multi_area_whitening. . . . 71
C.7 Normal_test. . . . 77
C.8 Whitening_Figures . . . 78

1

Introduction

In December 2001 Richard Reid, also known as the “Shoe Bomber”, attempted to ignite a bomb hidden in the soles of his shoes on a plane from Paris to Miami. Even though the bomb did not detonate, it is said that “his shoes were packed with enough plastic explosives to punch a hole through the side of the plane” [25]. Nowadays, when going through airport security, one needs to take off big shoes like hiking boots, which can hold up the line. Airports like Schiphol are already dealing with large queues, since the COVID-19 pandemic led to staff shortages, specifically security guards [19]. One way to increase the efficiency without compromising on safety could be the use of a shoe scanner.

Stage Gate 11 B.V. is a small company that innovates in airport security solutions, such as the Shoe Scanner. The Shoe Scanner uses differential reflectance spectroscopy in order to scan the outside of the shoe for relevant materials, such as explosives and narcotics. Passengers will not have to take off their shoes, as trying to hide illicit materials in shoes almost always leaves traces on the outside of the shoe, which can be detected [4]. The hyperspectral data then goes through a number of steps before classifying the substances on the shoe as safe or unsafe. This report will be focussing on one step of the preprocessing: whitening.

The whitening transformation transforms a random matrix into a matrix with expectation 0 and a covariance matrix equal to the identity matrix, i.e., it removes correlation between the pixels and each pixel has variance 1. In the case of working with real data, and thus not knowing the expectation, the whitening transformation transforms a data matrix into a matrix with row average 0 and a covariance matrix equal to the identity matrix. This leads to the first question: what does a data matrix look like before and after whitening? There are actually infinitely many possible whitening transformations, but in this thesis, six whitening transformations will be examined.

In the case of high contrast within a scan, the data can still be transformed with whitening, but the parts with high intensity will be transformed differently then the parts with low intensity. The difference in this transformation has consequences to the way in which the transformation of explosives and narcotics are being carried out. Because of this deviating transformation, the uniformity of the spectra cannot be guaranteed. It needs to be researched what exactly is the influence of a high contrast scenario on the whitening transformation. If indeed the uniformity of the spectra cannot be guaranteed, then multi-area whitening can be used, to avoid this. Multi-area whitening uses clustering to separate similar planes on the shoe. These planes are then whitened individually, after which the data is merged back together for classification. This hopefully leads to a better uniformity of the spectra of explosives and narcotics. When using multi-area whitening, much smaller data matrices are whitened than when an entire scan is whitened. Since a relation has been noticed between the number of pixels used in the whitening transformation and the signal-to-noise-ratio, this relation is also a topic that needs to be researched. And since the number of pixels being whitened matters, the cut-off criteria when using multi-area whitening will also be researched, i.e., how many pixels are needed at least in a cluster to perform properly.

Furthermore, when conducting the research, the question came up whether the data matrix that is used as input should be transposed or not, i.e., the question of should the pixels be viewed as the variables or the observations?

To summarize, the following research questions will be answered:

- What does a target spectrum look like before and after whitening?
- What is the relationship between the number of pixels used in the whitening transformation and the signal-to-noise-ratio?
- What is the influence of a high contrast scenario on the whitening transformation?
- How does 'normal' whitening compare to multi-area whitening?
- What is a suitable cut-off criterium for a cluster when using multi-area whitening?
- Should the data matrix used as input be transposed or not?

To be able to answer these questions in this thesis, first background information is given on hyperspectral imaging, covariance matrix estimation and the six whitening methods that were researched: PCA, standardized PCA, ZCA, standardized ZCA, Cholesky and CCA whitening in Chapter 2. Then, in Chapter 3, descriptions are given of the methodology and implementations that were used to give more insight into the whitening transformation, like the creating of synthetic data, the signal-to-noise ratio, Monte Carlo simulations and multi-area whitening. In Chapter 4, the results are discussed. In Chapter 5 conclusions to all results are drawn. And finally, in Chapter 6 possible complications will be discussed, and how the research in this thesis could be improved. Recommendations for further research will be given.

2

Background

2.1. Hyperspectral Imaging

Hyperspectral imaging is a technology that can be used for the identification of substances. Unpolarized, polychromatic light is sent out with a varying wavelength [9]. When reaching the object to be scanned, some of the light is absorbed, the rest is reflected. The intensity of the incoming light at every wavelength is measured. This way a spectrum is created, showing the intensity of the light measured at varying wavelength and thus at which wavelengths light is reflected or absorbed by the scanned substance.

There are two different ways to record this data, using multispectral imaging and hyperspectral imaging. In multispectral imaging, there are generally 3 to 10 spectral bands recorded [7]. In hyperspectral imaging, a larger number of spectral bands are recorded, namely hundreds, resulting in a continuous spectrum, see Figure 2.1. By analysing these spectral signatures, hyperspectral imaging enables us to identify and differentiate materials and substances, based on their unique spectral fingerprints. This technology can be used to scan a two-dimensional surface. This results in a detailed dataset known as a hyperspectral cube, where each two-dimensional pixel contains a whole spectrum.

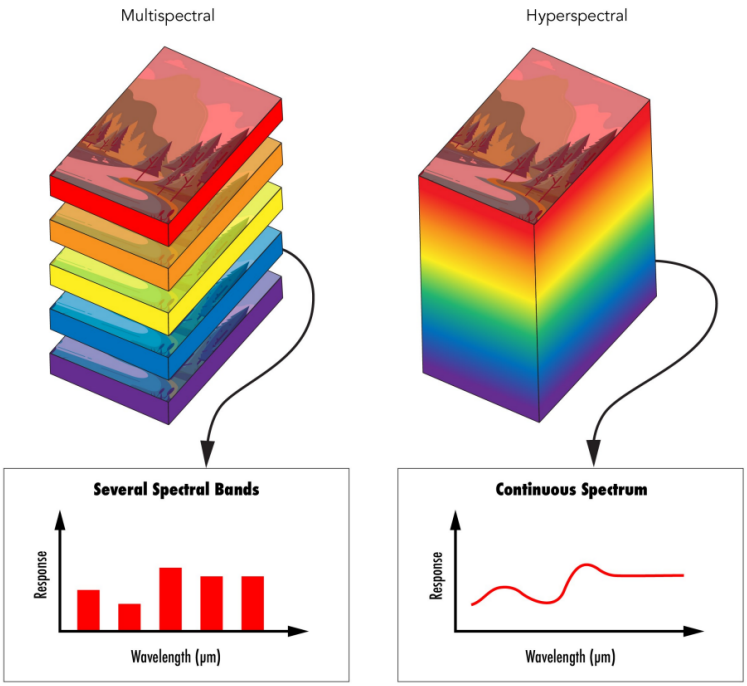


Figure 2.1: Multispectral and Hyperspectral imaging, figure taken from [5].

In our case, 548 spectral bands are used, ranging from 200 nm to 550 nm, i.e., ranging from UV to visible light, see Figure 2.2.

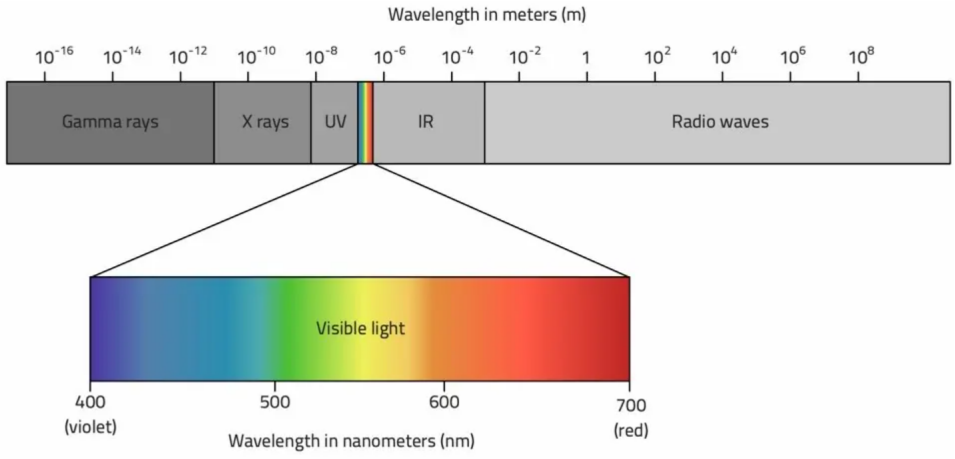


Figure 2.2: Electromagnetic spectrum, figure taken from [2].

Now, it is assumed that the hyperspectral cube is of size $x \times y \times z$, then the input image is of size $x \times y$ and the number of spectral bands is z. In order to easily process all this data, the hyperspectral cube is rearranged, such that it is two-dimensional. The pixels are rearranged from a two-dimensional picture into a line of length $x \cdot y$, meaning the data cube is now of size $(x \cdot y) \times z$. From here on out, the recorded data cube or image obtained from a scan will be called Im with size $n \times m$.

If $Im \in \mathbb{R}^{n \times m}$ is the input image of a scan, then it has n rows of variables, from which the row means need to be subtracted to centre the data, and m columns of observations. An entry of the input image is the intensities of a certain pixels at a certain wavelength. It can be chosen if n is the number of wavelengths that are used, which is 548 for all scans, or if n is the number of pixels in the scan, which varies per scan, i.e., it can be decided whether or not to transpose the input image. The effects of doing so will be discussed further into this report.

2.2. The Covariance Matrix

The true covariance matrix of a random matrix Y equals $\Sigma = \mathbb{E}\left[(Y - \mathbb{E}[Y])(Y - \mathbb{E}[Y])^T\right]$. If the distribution of Y is unknown, Σ must be estimated from the data. To do this, the input image Im is used. The covariance matrix of $Im \in \mathbb{R}^{n \times m}$ is estimated from the data using the empirical covariance matrix: $\hat{\Sigma} = \frac{1}{m-1} XX^T$, where X is row-wise centred. If $Im \in \mathbb{R}^{n \times m}$ is the input image, then $X_{i,k} = Im_{i,k} - \mu_i$, where μ_i is the mean of row i, i.e., $\mu_i = \frac{1}{m} \sum_{j=1}^{m} Im_{i,j}$, for $i = 1, \ldots, n$. The matrix $X \in \mathbb{R}^{n \times m}$ is considered a matrix consisting of m observations, i.e., column vectors $X = (X_1, X_2, \ldots, X_m)$ with $X_1, X_2, \ldots, X_m \in \mathbb{R}^n$.

Theorem 2.2.1. *Let $Im \in \mathbb{R}^{n \times m}$ be a random matrix with i.i.d columns Im_j, for $j = 1, \ldots, m$. Let $X \in \mathbb{R}^{n \times m}$ be a row-wise centred matrix with $X_{i,k} = Im_{i,k} - \mu_i$, where μ_i is the mean of row i of Im, i.e., $\mu_i = \frac{1}{m} \sum_{j=1}^{m} Im_{i,j}$, for $i = 1, \ldots, n$. Then the sample covariance matrix, $\hat{\Sigma} = \frac{1}{m-1} XX^T$, is an unbiased estimator of the covariance matrix of Im.*

Proof. Let $\underline{\mu} = (\mu, \mu, \ldots, \mu) \in \mathbb{R}^{n \times m}$ with $\mu = (\mu_1, \mu_2 \ldots, \mu_n)^T \in \mathbb{R}^n$, then

$$\hat{\Sigma} = \frac{1}{m-1}(Im - \underline{\mu})(Im - \underline{\mu})^T = \frac{1}{m-1} XX^T = \frac{1}{m-1} (X_1, X_2, \ldots, X_m) \begin{pmatrix} {X_1}^T \\ {X_2}^T \\ \vdots \\ {X_m}^T \end{pmatrix}$$

$$
\begin{aligned}
&= \frac{1}{m-1}\sum_{i=1}^{m}\begin{bmatrix} X_{i,1}X_{i,1} & X_{i,1}X_{i,2} & \dots & X_{i,1}X_{i,n} \\ X_{i,1}X_{i,2} & X_{i,2}X_{i,2} & \dots & X_{i,2}X_{i,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{i,1}X_{i,n} & X_{i,2}X_{i,n} & \dots & X_{i,n}X_{i,n} \end{bmatrix} = \frac{1}{m-1}\sum_{i=1}^{m} X_i {X_i}^T \\
&= \frac{1}{m-1}\sum_{i=1}^{m}(Im_i-\mu)(Im_i-\mu)^T = \frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T - \mu {Im_i}^T - Im_i\mu^T + \mu\mu^T\right) \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T - \mu {Im_i}^T - Im_i\mu^T\right) + \frac{m}{m-1}\mu\mu^T.
\end{aligned}
$$

Here, Im_i is the i^{th} column of Im. Now, $Im_i\mu^T$ and $\mu {Im_i}^T$ are looked at, and it is found that

$$\sum_{i=1}^{m}\left(Im_i\mu^T\right)^T = \sum_{i=1}^{m}\mu {Im_i}^T = \mu\sum_{i=1}^{m} {Im_i}^T = m\mu\mu^T.$$

So,

$$
\begin{aligned}
\hat{\Sigma} &= \frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T - \mu {Im_i}^T - Im_i\mu^T\right) + \frac{m}{m-1}\mu\mu^T \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T\right) - \frac{m}{m-1}\mu\mu^T - \left(\frac{m}{m-1}\mu\mu^T\right)^T + \frac{m}{m-1}\mu\mu^T = \frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T\right) - \frac{m}{m-1}\mu\mu^T.
\end{aligned}
$$

To prove that $\hat{\Sigma}$ is unbiased, it must be shown that its expectation equals Σ:

$$
\begin{aligned}
\mathbb{E}\left[\hat{\Sigma}\right] &= \mathbb{E}\left[\frac{1}{m-1}\sum_{i=1}^{m}\left(Im_i {Im_i}^T\right) - \frac{m}{m-1}\mu\mu^T\right] = \frac{1}{m-1}\sum_{i=1}^{m}\left(\mathbb{E}\left[Im_i {Im_i}^T\right]\right) - \frac{m}{m-1}\mathbb{E}\left[\mu\mu^T\right] \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(\mathrm{Var}(Im_i) + \mathbb{E}[Im_i]\mathbb{E}[Im_i]^T\right) - \frac{m}{m-1}\left(\mathrm{Var}(\mu) + \mathbb{E}[\mu]\mathbb{E}[\mu]^T\right)
\end{aligned}
$$

Now, use $\mathrm{Var}(Im_i) = \Sigma$, and thus $\mathrm{Var}(\mu) = \mathrm{Var}\left(\frac{1}{m}\sum_{i=1}^{m} Im_i\right) = \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}(Im_i) = \frac{1}{m}\Sigma$.

$$\mathbb{E}\left[\hat{\Sigma}\right] = \frac{m}{m-1}\left(\Sigma + \mu\mu^T\right) - \frac{m}{m-1}\left(\frac{1}{m}\Sigma + \mu\mu^T\right) = \Sigma.$$

□

The basis for the proof is taken from [1] and elaborated upon. In practice, the observations will not be independent and identically distributed random variables, but $\hat{\Sigma}$ will still be a good estimator, as long as the number of variables is smaller than the number of observations, but not when $\frac{m}{n}$ is large [18, 20, 22]. Covariance matrix estimation is a well researched problem within the field of statistics and will be discussed later.

Throughout this chapter, Σ will be written instead of $\hat{\Sigma}$ in the theoretical background, but all theorems also hold for $\hat{\Sigma}$, as is shown in Appendix A.

2.3. Whitening Tranformations

A whitening matrix $W \in \mathbb{R}^{n\times n}$ transforms the input random matrix $X \in \mathbb{R}^{n\times m}$ into a whitened matrix $Y \in \mathbb{R}^{n\times m}$ according to the formula: $Y = WX$. The whitened matrix has an expectation of 0 and a covariance matrix equal to the identity matrix. By removing the first and second order statistical structures, i.e., expectations, correlations and variances, higher order structures can be looked at for a better classification [15].

Definition 2.3.1. Let $X \in \mathbb{R}^{n\times m}$ be a row-wise centred data matrix with covariance matrix Σ. A whitening matrix of X is a matrix W such that $W^TW = \Sigma^{-1}$.

If W and Σ are invertible, this is equivalent to $W\Sigma W^T = I$, because

$$
\begin{aligned}
W^TW &= \Sigma^{-1} \\
\Sigma W^TW &= I \\
\Sigma W^T &= W^{-1} \\
W\Sigma W^T &= I.
\end{aligned}
$$

Theorem 2.3.2. *Let $X \in \mathbb{R}^{n\times m}$ be a row-wise centred random matrix with covariance matrix Σ, i.e., $\mathbb{E}[X] = 0$ and $\mathrm{Cov}(X) = \Sigma$. Let W be a whitening matrix of X. Then, the whitened matrix $Y = WX$ has expectation 0 and covariance matrix I.*

Proof. The following expectation and covariance are found for Y:

$$\begin{aligned}
\mathbb{E}[Y] &= \mathbb{E}[WX] = W\mathbb{E}[X] = 0,\\
\mathrm{Cov}(Y) &= \mathbb{E}\left[(Y-\mathbb{E}[Y])(Y-\mathbb{E}[Y])^T\right]\\
&= \mathbb{E}\left[(WX-\mathbb{E}[WX])(WX-\mathbb{E}[WX])^T\right]\\
&= \mathbb{E}\left[W(X-\mathbb{E}[X])(X-\mathbb{E}[X])^T W^T\right]\\
&= W\mathbb{E}\left[(X-\mathbb{E}[X])(X-\mathbb{E}[X])^T\right] W^T\\
&= W\Sigma W^T = I.
\end{aligned}$$

□

In reality, $\mathbb{E}[X]$ is unknown. Therefore, the empirical covariance matrix $\hat{\Sigma}_X$ of X is used, and thus also the corresponding whitening matrix $\hat{W}$ such that $\hat{W}^T\hat{W} = \hat{\Sigma}_X^{-1}$ is used. Then the empirical covariance matrix $\hat{\Sigma}_Y$ of Y will also be equal to the identity matrix:

$$\begin{aligned}
\hat{\Sigma}_Y &= \frac{1}{m-1}\left(\hat{W}X\right)\left(\hat{W}X\right)^T\\
&= \frac{1}{m-1}\hat{W}XX^T\hat{W}^T = \hat{W}\hat{\Sigma}_X\hat{W}^T = I.
\end{aligned}$$

As mentioned earlier, write Σ instead of $\hat{\Sigma}$ will be written throughout this chapter. But as long as the corresponding whitening matrix $\hat{W}$ and other corresponding matrices, all estimated from $\hat{\Sigma}$, are used, all theorems also hold for $\hat{\Sigma}$. This includes matrices such as $\hat{V}, \hat{D}, \hat{S}, \hat{P}, \hat{R}, \hat{\Phi}, \hat{\Psi}$ that appear later in this chapter.

Since the covariance matrix is a real and symmetric matrix, an eigendecomposition can be performed: $\Sigma = VDV^T$, where V is a orthogonal matrix with the orthonormal eigenvectors of Σ as its columns, and D a diagonal matrix with the corresponding eigenvalues as its diagonal values. The eigenvalues can be put in descending order, such that $D_{i,i} \geq D_{i+1,i+1}$.

Theorem 2.3.3. *If $W = U_1\Sigma^{-1/2} = U_1VD^{-1/2}V^T$ with U_1 any orthogonal matrix of size $n \times n$, then W is a whitening matrix.*

Proof. U_1 and V are orthogonal, i.e., $U_1^T = U_1^{-1}$ and $V^T = V^{-1}$. This is used to find

$$\begin{aligned}
W^TW &= \left(V^T\right)^T D^{-1/2}V^T U_1^T U_1 V D^{-1/2}V^T\\
&= VD^{-1/2}V^T VD^{-1/2}V^T\\
&= VD^{-1/2}D^{-1/2}V^T\\
&= VD^{-1}V^T = \Sigma^{-1},\\
W\Sigma W^T &= U_1\Sigma^{-1/2}\Sigma\Sigma^{-1/2}U_1^T = U_1U_1^T = I.
\end{aligned}$$

□

Thus, the whitening transformation combines multivariate rescaling by $\Sigma^{-1/2}$ and rotation by U_1. Different choices for the orthogonal matrix U_1 correspond to different whitening methods. This is called rotational freedom in whitening [13]. There are infinitely many whitening matrices that satisfy Definition 2.3.1.

Another possibility is to decompose the covariance matrix Σ into the correlation matrix P and diagonal covariance matrix S, such that $\Sigma = S^{1/2}PS^{1/2}$. The input matrix $X \in \mathbb{R}^{n\times m}$ can be interpreted as a matrix of row vectors with mean zero, then $X = (X_1, X_2, \ldots, X_n)^T$ with $X_1, X_2, \ldots, X_n \in \mathbb{R}^{1\times m}$ and $\mathbb{E}[X_i] = 0$ for $i = 1, \ldots, n$. The following diagonal covariance matrix S and correlation matrix P can be found:

$$S = \begin{bmatrix} \sigma_{X_1}^2 & 0 & \cdots & 0\\ 0 & \sigma_{X_1}^2 & & \vdots\\ \vdots & & \ddots & 0\\ 0 & \cdots & 0 & \sigma_{X_n}^2 \end{bmatrix} \text{ and } P = \begin{bmatrix} \frac{\mathrm{Cov}(X_1,X_1)}{\sigma_{X_1}\sigma_{X_1}} & \frac{\mathrm{Cov}(X_1,X_2)}{\sigma_{X_1}\sigma_{X_2}} & \cdots & \frac{\mathrm{Cov}(X_1,X_n)}{\sigma_{X_1}\sigma_{X_n}}\\ \frac{\mathrm{Cov}(X_1,X_2)}{\sigma_{X_1}\sigma_{X_2}} & \frac{\mathrm{Cov}(X_2,X_2)}{\sigma_{X_2}\sigma_{X_2}} & \cdots & \frac{\mathrm{Cov}(X_2,X_n)}{\sigma_{X_2}\sigma_{X_n}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\mathrm{Cov}(X_1,X_n)}{\sigma_{X_1}\sigma_{X_n}} & \frac{\mathrm{Cov}(X_2,X_n)}{\sigma_{X_2}\sigma_{X_n}} & \cdots & \frac{\mathrm{Cov}(X_n,X_n)}{\sigma_{X_n}\sigma_{X_n}} \end{bmatrix}.$$

Note that the diagonal elements of P, $\text{cor}(X_i, X_i) = \frac{\text{Cov}(X_i, X_i)}{\sigma_{X_i}\sigma_{X_i}}$ should equal 1. Using this, one finds that

$$\Sigma = S^{1/2}PS^{1/2} = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_1, X_2) & \text{Cov}(X_2, X_2) & \dots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_1, X_n) & \text{Cov}(X_2, X_n) & \dots & \text{Cov}(X_n, X_n) \end{bmatrix}.$$

Theorem 2.3.4. *If $W = U_2 P^{-1/2} S^{-1/2}$ with U_2 any orthogonal matrix of size $n \times n$, then W is a whitening matrix.*

Proof. Remember that P and S are symmetric and U_2 is orthogonal, i.e., $U_2^T = U_2^{-1}$. Using this, one finds that

$$\begin{aligned} W^T W &= S^{-1/2}P^{-1/2}U_2^T U_2 P^{-1/2}S^{-1/2} = S^{-1/2}P^{-1/2}P^{-1/2}S^{-1/2} = S^{-1/2}P^{-1}S^{-1/2} = \Sigma^{-1}, \\ W\Sigma W^T &= U_2 P^{-1/2}S^{-1/2}S^{1/2}PS^{1/2}S^{-1/2}P^{-1/2}U_2^T = U_2 P^{-1/2}PP^{-1/2}U_2^T = U_2 U_2^T = I. \end{aligned}$$

□

The matrix P is also a real symmetric matrix, thus an eigendecomposition can be performed on $P = G\Theta G^T$. "In this view, with $W = U_2 G\Theta^{-1/2}G^T S^{-1/2}$, the variables are first scaled by the square root of the diagonal variance matrix, then rotated by G^T , then scaled again by the square root of the eigenvalues of the correlation matrix, and possibly rotated once more (depending on the choice of U_2)." [13]

For a certain whitening transformation, both W from Theorem 2.3.3 and 2.3.4 should obviously be the same, this gives a relation between the rotation matrices U_1 and U_2:

$$\begin{aligned} W &= U_1\Sigma^{-1/2} = U_2 P^{-1/2}S^{-1/2}, \\ U_1 &= U_2 P^{-1/2}S^{-1/2}\Sigma^{1/2} = U_2 R. \end{aligned}$$

Here the matrix $R = P^{-1/2}S^{-1/2}\Sigma^{1/2}$ is orthogonal:

$$\begin{aligned} RR^T &= P^{-1/2}S^{-1/2}\Sigma^{1/2}\Sigma^{1/2}S^{-1/2}P^{-1/2} = P^{-1/2}S^{-1/2}\Sigma S^{-1/2}P^{-1/2} \\ &= P^{-1/2}S^{-1/2}S^{1/2}PS^{1/2}S^{-1/2}P^{-1/2} = P^{-1/2}PP^{-1/2} = I, \\ R^T R &= \Sigma^{1/2}S^{-1/2}P^{-1/2}P^{-1/2}S^{-1/2}\Sigma^{1/2} = \Sigma^{1/2}\left(S^{-1/2}P^{-1}S^{-1/2}\right)\Sigma^{1/2} = \Sigma^{1/2}\Sigma^{-1}\Sigma^{1/2} = I, \\ R &= \left(R^T\right)^{-1} = P^{1/2}S^{1/2}\Sigma^{-1/2}. \end{aligned}$$

Now, like Kessy et al., 2018 [13], the cross-covariance Φ and cross-correlation Ψ matrices between the whitened matrix $Y = WX$ and the original data centred matrix X can be examined. A link to the rotation matrices U_1 and U_2 is found:

$$\begin{aligned} \Phi &= \text{Cov}(WX, X) = \mathbb{E}\left[(WX - \mathbb{E}[WX])(X - \mathbb{E}[X])^T\right] = W\Sigma = U_1\Sigma^{-1/2}\Sigma = U_1\Sigma^{1/2}, \\ \Psi &= \text{cor}(WX, X) = \Phi S^{-1/2} = U_1\Sigma^{1/2}S^{-1/2} = U_2 R\Sigma^{1/2}S^{-1/2} = U_2 P^{1/2}S^{1/2}\Sigma^{-1/2}\Sigma^{1/2}S^{-1/2} = U_2 P^{1/2}S^{1/2}S^{-1/2} = U_2 P^{1/2}. \end{aligned}$$

2.3.1. Principal Component Analysis (PCA)

PCA is an orthogonal linear transformation, i.e., it preserves the inner product, scalar multiplication and vector addition. It transforms a column-wise centred dataset $C \in \mathbb{R}^{m \times n}$, with each of the m rows representing an observation and each of the n columns representing a variable, to a new coordinate system C_{PCA}, where the data is projected upon the principal components. The first coordinate is the projection upon the first principal component, the second coordinate the projection upon the second principal component, etc. The principal components are orthonormal vectors that point in the direction of the greatest variance of the dataset, with the first principal component accounting for the most of the variance, the second principal component accounting for the most of the variance after the first, etc. Since the input image is a row-wise centred dataset $X \in \mathbb{R}^{n \times m}$, it is clear that $C = X^T$.

To have the first principal component p_1 pointing in the direction of the greatest variance, it has to satisfy the following equation:

$$\begin{aligned} p_1 &= \underset{||p||=1}{\arg\max}\left\{||Cp||^2\right\} = \underset{||p||=1}{\arg\max}\left\{p^T C^T C p\right\} \\ &= \underset{||p||=1}{\arg\max}\left\{p^T X X^T p\right\} = \underset{||p||=1}{\arg\max}\left\{p^T \hat{\Sigma} p\right\} \end{aligned}$$

$$= \quad \arg\max\left\{\frac{p^T\hat{\Sigma}p}{p^Tp}\right\}. \tag{2.1}$$

This equals the eigenvector corresponding to the largest eigenvalue of $\hat{\Sigma}$. The i^{th} principal component also needs to satisfy equation 2.1, with the additional constraint of being orthogonal to the first $i-1$ principal components. This turns out to be the eigenvector corresponding to the i^{th} largest eigenvalue of $\hat{\Sigma}$. The factor of $\frac{1}{m-1}$ can be ignored, since it does not affect the argmax. Note that if an eigenvalue has a multiplicity higher than 1, say m, i.e., it corresponds to m eigenvectors, say $p_i,\ldots,p_{i+m}$, then $p_i,\ldots,p_{i+m}$ are interchangeable and make up the next m principal components.

Theorem 2.3.5. *The principal components of the column-wise centred dataset $X^T \in \mathbb{R}^{m\times n}$ equal the eigenvectors of the empirical covariance matrix $\hat{\Sigma} = \frac{1}{m-1}XX^T$, with the first principal component being equal to the eigenvector that corresponds to the largest eigenvalue of $\hat{\Sigma}$, the second principal component being equal to the eigenvector that corresponds to the second largest eigenvalue, etc.*

Proof. A proof by induction is used. First, there will be a proof that the theorem holds for the first principal component, for this the proof by Severn, K. (2023) [24] is followed. The aim is to maximize $p^T\hat{\Sigma}p$ subject to $||p|| = 1$, i.e., $p^Tp = 1$ or $1-p^Tp = 0$. This can be achieved using Lagrange multipliers, differentiating with respect to p and λ and setting both to 0, in order to find the maximum:

$$\begin{aligned}
\mathscr{L}(p,\lambda) &= p^T\hat{\Sigma}p + \lambda(1-p^Tp),\\
\nabla_{p,\lambda}\mathscr{L}(p,\lambda) &= \left(\frac{\partial\mathscr{L}}{\partial p},\frac{\partial\mathscr{L}}{\partial\lambda}\right)\\
&= \left(2\hat{\Sigma}p - 2\lambda p, 1-p^Tp\right),\\
\nabla_{p,\lambda}\mathscr{L}(p,\lambda) = 0 &\Leftrightarrow \begin{cases} 2\hat{\Sigma}p - 2\lambda p = 0\\ 1-p^Tp = 0\end{cases},\\
&= \begin{cases}\hat{\Sigma}p = \lambda p\\ p^Tp = 1\end{cases}.
\end{aligned}$$

Thus, p is an eigenvector of $\hat{\Sigma}$, subject to $p^Tp = 1$, corresponding to eigenvalue λ. If this is substituted back into $\mathscr{L}$, one gets:

$$\mathscr{L}(p,\lambda) = p^T\hat{\Sigma}p + \lambda(1-p^Tp) = p^T\hat{\Sigma}p = \lambda p^Tp = \lambda.$$

To maximize this, λ needs to be the largest eigenvalue of $\hat{\Sigma}$. Thus, it is indeed found that the first principal component, p_1, is equal to the unit eigenvector that corresponds to the largest eigenvalue of $\hat{\Sigma}$.
Now, it is assumed that the first $j-1$ principal components equal the unit eigenvectors that correspond to the $j-1$ largest eigenvalues of $\hat{\Sigma}$, and proven this also holds for the j^{th} principal component. Again the aim is to maximize $p_j^T\hat{\Sigma}p_j$, subject not only to $||p_j|| = 1$, but also to $p_j^Tp_i = 0$ for $i = 1,\ldots,j-1$, because the j^{th} principal component must be orthogonal to the first $j-1$ principal components. Again, use Lagrange multipliers:

$$\begin{aligned}
\mathscr{L}(p_j,\lambda,\mu_1,\ldots,\mu_{j-1}) &= p_j^T\hat{\Sigma}p_j + \lambda(1-p_j^Tp_j) + \sum_{i=1}^{j-1}\mu_i(0-p_j^Tp_i),\\
\nabla_{p_j,\lambda,\mu_1,\ldots,\mu_{j-1}}\mathscr{L}(p_j,\lambda,\mu_1,\ldots,\mu_{j-1}) &= \left(\frac{\partial\mathscr{L}}{\partial p_j},\frac{\partial\mathscr{L}}{\partial\lambda},\frac{\partial\mathscr{L}}{\partial\mu_1},\ldots,\frac{\partial\mathscr{L}}{\partial\mu_{j-1}}\right)\\
&= \left(2\hat{\Sigma}p_j - 2\lambda p_j - \sum_{i=1}^{j-1}\mu_ip_i, 1-p_j^Tp_j, -p_j^Tp_1,\ldots,-p_j^Tp_{j-1}\right),\\
\nabla_{p_j,\lambda,\mu_1,\ldots,\mu_{j-1}}\mathscr{L}(p_j,\lambda,\mu_1,\ldots,\mu_{j-1}) = 0 &\Leftrightarrow \begin{cases} 2\Sigma p_j - 2\lambda p_j + \sum_{i=1}^{j-1}\mu_ip_i = 0\\ 1-p_j^Tp_j = 0\\ p_j^Tp_i = 0 \text{ for all } i = 1,\ldots,j-1\end{cases}.
\end{aligned}$$

Take the first of the three equations and left multiply by p_k^T for some $k \in \{1,\ldots,j-1\}$, where p_k is a unit eigenvector of $\hat{\Sigma}$ by assumption. Note that all p_i are mutually orthogonal, i.e., $p_k^Tp_i = \begin{cases}0 \text{ if } k\neq i\\ 1 \text{ if } k = i\end{cases}$.
One finds that

$$p_j^Tp_k = 0 \quad\Leftrightarrow\quad p_k^Tp_j = 0,$$

$$\begin{aligned} 0 &= 2p_k^T\hat{\Sigma}p_j - 2\lambda p_k^T p_j + \sum_{i=1}^{j-1}\mu_i p_k^T p_i \\ &= 2\left(\hat{\Sigma}p_k\right)^T p_j - 2\lambda p_k^T p_j + \mu_k p_k^T p_k \\ &= 2\left(\lambda_k p_k\right)^T p_j - 2\left(\lambda_k p_k\right)^T p_j + \mu_k = \mu_k. \end{aligned}$$

It is obtained that $\mu_k = 0$ for all $k = 1,\ldots,j-1$. When this is filled in at $\frac{\partial\mathscr{L}}{\partial p_j}$, it is then found that $\hat{\Sigma}p_j = \lambda p_j$. Thus, if the aim is to maximize this given that p_j should be orthogonal to $p_1,\ldots,p_{j-1}$, then p_j must be equal to the eigenvector of $\hat{\Sigma}$ that corresponds to the j^{th} largest eigenvalue. □

For PCA whitening, choose U_1 from Theorem 2.3.3 to be V^T, i.e., $W = V^T\Sigma^{-1/2} = V^T VD^{-1/2}V^T = D^{-1/2}V^T$. The matrix V^T rotates the data, such that the points are projected upon the principal components. This means that the data is now decorrelated, but not yet with variance 1. This is because

$$\begin{aligned} \text{Cov}\left(V^T X\right) &= \mathbb{E}\left[\left(V^T X - \mathbb{E}\left[V^T X\right]\right)\left(V^T X - \mathbb{E}\left[V^T X\right]\right)^T\right] \\ &= \mathbb{E}\left[\left(V^T X - V^T\mathbb{E}[X]\right)\left(V^T X - V^T\mathbb{E}[X]\right)^T\right] \\ &= V^T\mathbb{E}\left[\left(X - \mathbb{E}[X]\right)\left(X - \mathbb{E}[X]\right)^T\right]V \\ &= V^T\Sigma V = V^T VDV^T V = D. \end{aligned}$$

To get the variances to 1, multiply by $D^{-1/2}$ and find

$$\begin{aligned} \text{Cov}(Y) &= \text{Cov}\left(D^{-1/2}V^T X\right) \\ &= \mathbb{E}\left[\left(D^{-1/2}V^T X - \mathbb{E}\left[D^{-1/2}V^T X\right]\right)\left(D^{-1/2}V^T X - \mathbb{E}\left[D^{-1/2}V^T X\right]\right)^T\right] \\ &= \mathbb{E}\left[\left(D^{-1/2}V^T X - D^{-1/2}V^T\mathbb{E}[X]\right)\left(D^{-1/2}V^T X - D^{-1/2}V^T\mathbb{E}[X]\right)^T\right] \\ &= D^{-1/2}V^T\mathbb{E}\left[\left(X - \mathbb{E}[X]\right)\left(X - \mathbb{E}[X]\right)^T\right]VD^{-1/2} \\ &= D^{-1/2}V^T\Sigma VD^{-1/2} = D^{-1/2}V^T VDV^T D^{-1/2}V = D^{-1/2}DD^{-1/2} = I. \end{aligned}$$

Thus, when applying PCA whitening, the data is first projected upon the principal components to decorrelate it, and then divided by the square root of the eigenvalue to get the variances to equal 1. This gives maximal $\phi_i = \sum_{j=1}^n \phi_{i,j}^2 = \sum_{j=1}^n \text{Cov}(WX_i, X_j)^2$.

Theorem 2.3.6. *ϕ_i with $i \in \{1,\ldots,n\}$ are successively maximized for $W = W_{PCA} = D^{-1/2}V^T$, with $\phi_i \geq \phi_{i+1}$.*

Proof. First, maximize ϕ_1, then ϕ_2 , etc. up to ϕ_n. This ensures that $\phi_i \geq \phi_{i+1}$. First, look at the general ϕ_i:

$$\begin{aligned} \phi_i &= \sum_{j=1}^n \phi_{i,j}^2 = \left(\Phi\Phi^T\right)_{i,i}, \\ \left(\phi_1,\ldots,\phi_n\right)^T &= \text{diag}\left(\Phi\Phi^T\right) = \text{diag}\left(U_1\Sigma^{1/2}\left(U_1\Sigma^{1/2}\right)^T\right) = \text{diag}\left(U_1\Sigma^{1/2}\Sigma^{1/2}U_1^T\right) \\ &= \text{diag}\left(U_1\Sigma U_1^T\right) = \text{diag}\left(U_1 VDV^T U_1^T\right) = \text{diag}\left(QDQ^T\right), \end{aligned}$$

where $Q = U_1 V \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, because U_1 and V are orthogonal matrices. D was a diagonal matrix with the eigenvalues of Σ on its diagonal, in descending order. Thus, it is obtained that:

$$\begin{aligned} \phi_i &= \left(QDQ^T\right)_{i,i} = \sum_{k=1}^n (QD)_{i,k}\left(Q^T\right)_{k,i} = \sum_{k=1}^n\left(\sum_{j=1}^n Q_{i,j}D_{j,k}\right)Q_{i,k}, \\ D_{j,k} &= 0 \text{ if } j \neq k, \\ \phi_i &= \sum_{k=1}^n Q_{i,k}D_{k,k}Q_{i,k} = \sum_{k=1}^n D_{k,k}Q_{i,k}^2. \end{aligned}$$

Since Q is orthogonal, every row and column is orthonormal, thus $\sum_{k=1}^n Q_{i,k}Q_{j,k} = 0$ for $i \neq j$ and 1 for $i = j$, and $\sum_{k=1}^n Q_{k,i}Q_{k,j} = 0$ for $i \neq j$ and 1 for $i = j$. Furthermore, it is known that $D_{1,1} \geq \ldots, \geq D_{n,n}$. Therefore, ϕ_1 is maximized when $Q_{1,1} = 1$ and $Q_{1,k} = 0$ for all other k. This gives $\phi_1 = D_{1,1}$. So Q has a one as it first entry, and

zeros in the rest of the first row. Because of Q's orthogonality, the rest of the first column should also contain zeros. If now ϕ_2 is maximized, it is found that $\phi_2 = \sum_{k=2}^{n} D_{k,k} Q_{2,k}^2$, since $Q_{1,k} = 0$. Again, this is maximized when $Q_{2,2} = 1$ and $Q_{2,k} = 0$ for all other k. This gives $\phi_2 = D_{2,2}$. Continue this pattern, and one finds that the ϕ_i are maximized if $Q = I$. They are indeed in descending order, since $\phi_i = D_{i,i} \geq D_{i+1,i+1} = \phi_{i+1}$. The result are that $U_1 = QV^T = V^T$ and $W = U_1 \Sigma^{-1/2} = V^T V D^{-1/2} V^T = D^{-1/2} V^T = W_{PCA}$. □

2.3.2. Standardized PCA

For standardized PCA whitening, choose U_2 from Theorem 2.3.4 to be G^T, i.e., $W = G^T P^{-1/2} S^{-1/2}$ $= G^T G \Theta^{-1/2} G^T S^{-1/2} = \Theta^{-1/2} G^T S^{-1/2}$. Similar to PCA, standardized PCA maximizes the $\psi_i = \sum_{j=1}^{n} \psi_{i,j}^2$ $= \sum_{j=1}^{n} \text{cor}(WX_i, X_j)^2$.

Theorem 2.3.7. *ψ_i with $i \in \{1, \ldots, n\}$ are successively maximized for $W = W_{PCA-std} = \Theta^{-1/2} G^T S^{-1/2}$, with $\psi_i \geq \psi_{i+1}$.*

Proof. First maximize ψ_1, then ψ_2 , etc. up to ψ_n. This ensures that $\psi_i \geq \psi_{i+1}$. The general ψ_i are first considered:

$$\begin{aligned} \psi_i &= \sum_{j=1}^{n} \psi_{i,j}^2 = \left(\Psi \Psi^T\right)_{i,i}, \\ \left(\psi_1, \ldots, \psi_n\right)^T &= \text{diag}\left(\Psi \Psi^T\right) = \text{diag}\left(U_2 P^{1/2} \left(U_2 P^{1/2}\right)^T\right) = \text{diag}\left(U_2 P^{1/2} P^{1/2} U_2^T\right) \\ &= \text{diag}\left(U_2 P U_2^T\right) = \text{diag}\left(U_2 G \Theta G^T U_2^T\right) = \text{diag}\left(Q \Theta Q^T\right), \end{aligned}$$

where $Q = U_2 G$ is an orthogonal matrix, because U_2 and G are orthogonal matrices. Θ is a diagonal matrix with the eigenvalues of P on its diagonal, in descending order. Just as in the proof of Theorem 2.3.6, it is found that $Q = I$ and therefore $U_2 = G^T$ and $W = \Theta^{-1/2} G^T S^{-1/2} = W_{PCA-std}$. □

2.3.3. Zero-phase Component Analysis (ZCA)

For Zero-phase Component Analysis, or ZCA whitening, choose U_1 from Theorem 2.3.3 to be the identity matrix, i.e., $W = \Sigma^{-1/2} = V D^{-1/2} V^T$. This has the same first two steps as PCA, rotation by V^T and scaling by $D^{-1/2}$, but then the data is rotated back by V to the original coordinate system. Because of this, ZCA is the "closest" to the original data in the least squares sense, using the Frobenius norm [13].

Theorem 2.3.8. *$||X - WX||_F^2$ is minimized for $W = W_{ZCA} = \hat{\Sigma}^{-1/2}$, given that X is a data centred matrix and WX is a whitened matrix.*

Proof. The proof is taken from the supplementary material from [27]. First, show that for any matrix $B \in \mathbb{R}^{n \times m}$ it holds that $||B||_F^2 = tr(BB^T)$:

$$\begin{aligned} ||B||_F^2 &= \sum_{i=1}^{n} \sum_{j=1}^{m} B_{i,j}^2, \\ \left(BB^T\right)_{i,k} &= \sum_{j=1}^{m} B_{i,j} \left(B^T\right)_{j,k} = \sum_{j=1}^{m} B_{i,j} B_{k,j}, \\ tr(BB^T) &= \sum_{i=1}^{n} \left(BB^T\right)_{i,i} = \sum_{i=1}^{n} \sum_{j=1}^{m} B_{i,j} B_{i,j} = ||B||_F^2. \end{aligned}$$

Now, remember the following properties: $\hat{\Sigma} = \frac{1}{m-1} XX^T = \hat{V} \hat{D} \hat{V}^T$, $W \hat{\Sigma} W^T = I$ and $W = U_1 \hat{\Sigma}^{-1/2}$ where $U_1^T = U_1^{-1}$. This is used to calculate the minimum of $||X - WX||_F^2$:

$$\begin{aligned} \min_W ||X - WX||_F^2 &= \min_W tr\left((X - WX)(X - WX)^T\right) \\ &= \min_W tr\left(XX^T - WXX^T - XX^T W^T + WXX^T W^T\right) \\ &= \min_W tr\left(XX^T - WXX^T - \left(WXX^T\right)^T + WXX^T W^T\right) \\ &= \min_W tr\left(XX^T\right) - 2tr\left(WXX^T\right) + tr\left(WXX^T W^T\right) \end{aligned}$$

$$\begin{aligned} &= \min_W (m-1)tr\left(\hat{\Sigma}\right) - 2(m-1)tr\left(W\hat{\Sigma}\right) + (m-1)tr\left(W\hat{\Sigma}W^T\right) \\ &= \min_W (m-1)tr\left(\hat{\Sigma}\right) - 2(m-1)tr\left(W\hat{\Sigma}\right) + (m-1)tr\left(I\right). \end{aligned}$$

Only $tr\left(W\hat{\Sigma}\right)$ depends on W, so minimizing the result above w.r.t. W is the same as maximizing $tr(W\hat{\Sigma})$ with respect to W, and therefore to problem is equivalent to

$$\begin{aligned} \max_W tr(W\hat{\Sigma}) &= \max_{\substack{W=U_1\hat{\Sigma}^{-1/2} \\ U_1^T=U_1^{-1}}} tr(U_1\hat{\Sigma}^{-1/2}\hat{\Sigma}) = \max_{\substack{U_1 \\ U_1^T=U_1^{-1}}} tr(U_1\hat{\Sigma}^{1/2}) \\ &= \max_{\substack{U_1 \\ U_1^T=U_1^{-1}}} tr\left(U_1\hat{V}\hat{D}^{1/2}\hat{V}^T\right) = \max_{\substack{U_1 \\ U_1^T=U_1^{-1}}} tr\left(\hat{D}^{1/2}\hat{V}^TU_1\hat{V}\right) \\ &= \max_{\substack{Q=\hat{V}^TU_1\hat{V} \\ U_1^T=U_1^{-1}}} tr\left(\hat{D}^{1/2}Q\right) = \sum_{i=1}^n \hat{D}_{i,i}^{1/2}Q_{i,i}, \end{aligned}$$

where the fact that $tr(AB) = tr(BA)$ if A and B are square matrices was used, which they are, since $A = U_1\hat{V}$ and $B = \hat{D}^{1/2}\hat{V}^T$. Furthermore, the last step can be done, because $\hat{D}$ is a diagonal matrix. Since U_1 and $\hat{V}$ are real orthogonal matrices, Q is also a real orthogonal matrix. Therefore, its entries cannot be bigger than one, including the diagonal entries, i.e., $Q_{i,i} \leq 1$. The following inequality follows from this fact:

$$\max_W tr(W\hat{\Sigma}) = \sum_{i=1}^n \hat{D}_{i,i}^{1/2}Q_{i,i} \leq \sum_{i=1}^n \hat{D}_{i,i}^{1/2},$$

with equality if and only if all diagonal entries of Q equal 1. Since Q is orthogonal, (i.e., with orthonormal rows and columns,) this can only happen when $Q = I$. Because $Q = \hat{V}^TU_1\hat{V} = I$, one gets $\hat{V}\hat{V}^TU_1\hat{V}\hat{V}^T = \hat{V}\hat{V}^T \Leftrightarrow U_1 = I$. So the whitening matrix W that uniquely minimizes $||X - WX||_F^2$, is $W = U_1\hat{\Sigma}^{-1/2} = \hat{\Sigma}^{-1/2}$. □

2.3.4. Standardized ZCA

For standardized ZCA whitening, choose U_2 from Theorem 2.3.4 to be the identity matrix, i.e., $W = P^{-1/2}S^{-1/2}$. Because of this, ZCA is the "closest" to the standardized data $S^{-1/2}X$ in the least squares sense, using the Frobenius norm [13].

Theorem 2.3.9. *$||WX - \hat{S}^{-1/2}X||_F^2$ is minimized for $W = W_{ZCA-std} = \hat{P}^{-1/2}\hat{S}^{-1/2}$, given that X is a data centred matrix and WX is a whitened matrix.*

Proof. Remember the following properties: $\hat{\Sigma} = \frac{1}{m-1}XX^T = \hat{S}^{1/2}\hat{P}\hat{S}^{1/2}$, $W\hat{\Sigma}W^T = I$ and $W = U_2\hat{P}^{-1/2}\hat{S}^{-1/2}$ where $U_2^T = U_2^{-1}$. This is used to calculate the minimum of $||WX - \hat{S}^{-1/2}X||_F^2$:

$$\begin{aligned} \min_W ||WX - \hat{S}^{-1/2}X||_F^2 &= \min_W tr\left((WX - \hat{S}^{-1/2}X)(WX - \hat{S}^{-1/2}X)^T\right) \\ &= \min_W tr\left(WXX^TW^T - WXX^T\hat{S}^{-1/2} - \hat{S}^{-1/2}XX^TW^T + \hat{S}^{-1/2}XX^T\hat{S}^{-1/2}\right) \\ &= \min_W tr\left(WXX^TW^T - WXX^T\hat{S}^{-1/2} - \left(WXX^T\hat{S}^{-1/2}\right)^T + \hat{S}^{-1/2}XX^T\hat{S}^{-1/2}\right) \\ &= \min_W tr\left(WXX^TW^T\right) - 2tr\left(WXX^T\hat{S}^{-1/2}\right) + tr\left(\hat{S}^{-1/2}XX^T\hat{S}^{-1/2}\right) \\ &= \min_W (m-1)tr\left(W\hat{\Sigma}W^T\right) - 2(m-1)tr\left(W\hat{\Sigma}\hat{S}^{-1/2}\right) + (m-1)tr\left(\hat{S}^{-1/2}\hat{\Sigma}S^{-1/2}\right) \\ &= \min_W (m-1)tr\left(I\right) - 2(m-1)tr\left(W\hat{S}^{1/2}\hat{P}\right) + (m-1)tr\left(\hat{P}\right). \end{aligned}$$

Only $tr\left(W\hat{S}^{1/2}\hat{P}\right)$ depends on W, so minimizing the result above w.r.t. W is the same as maximizing $tr\left(W\hat{S}^{1/2}\hat{P}\right)$ with respect to W.

$$\begin{aligned} \max_W tr\left(W\hat{S}^{1/2}\hat{P}\right) &= \max_{\substack{W=U_2\hat{P}^{-1/2}\hat{S}^{-1/2} \\ U_2^T=U_2^{-1}}} tr(U_2\hat{P}^{-1/2}\hat{S}^{-1/2}\hat{S}^{1/2}\hat{P}) = \max_{\substack{U_2 \\ U_2^T=U_2^{-1}}} tr(U_2\hat{P}^{-1/2}\hat{P}) = \max_{\substack{U_2 \\ U_2^T=U_2^{-1}}} tr(U_2\hat{P}^{1/2}) \\ &= \max_{\substack{U_2 \\ U_2^T=U_2^{-1}}} tr\left(U_2\hat{G}\hat{\Theta}^{1/2}\hat{G}^T\right) = \max_{\substack{U_2 \\ U_2^T=U_2^{-1}}} tr\left(\hat{\Theta}^{1/2}\hat{G}^TU_2\hat{G}\right) \end{aligned}$$

$$= \max_{Q=\hat{G}^T U_2 \hat{G}} tr\left(\hat{\Theta}^{1/2} Q\right) = \sum_{i=1}^{n} \hat{\Theta}_{i,i}^{1/2} Q_{i,i}$$

Where the fact that $tr(AB) = tr(BA)$ if A and B are square matrices was used, which they are, since $A = U_2\hat{G}$ and $B = \hat{\Theta}^{1/2}\hat{G}$. Furthermore, the last step can be done, because $\hat{\Theta}$ is a diagonal matrix. Since U_2 and $\hat{G}$ are real orthogonal matrices, Q is also a real orthogonal matrix. Therefore, its entries cannot be bigger than one, including the diagonal entries, i.e., $Q_{i,i} \leq 1$. The following inequality follows from this fact:

$$\max_W tr\left(W\hat{S}^{1/2}\hat{P}\right) = \sum_{i=1}^{n} \hat{\Theta}_{i,i}^{1/2} Q_{i,i} \leq \sum_{i=1}^{n} \hat{\Theta}_{i,i}^{1/2},$$

with equality if and only if all diagonal entries of Q equal 1. Since Q is orthogonal, (i.e., with orthonormal rows and columns,) this can only happen when $Q = I$. Because $Q = \hat{G}^T U_2 \hat{G} = I$, one gets $\hat{G}\hat{G}^T U_2 \hat{G}\hat{G}^T = \hat{G}\hat{G}^T \Leftrightarrow U_2 = I$. So the whitening matrix W that uniquely minimizes $||WX - \hat{S}^{-1/2}X||_F^2$, is $W = U_2\hat{P}^{-1/2}\hat{S}^{-1/2} = \hat{P}^{-1/2}\hat{S}^{-1/2}$. □

2.3.5. Canonical Correlation Analysis (CCA)

CCA is an algorithm used to examine the linear relationship between two datasets. It finds canonical directions $\alpha_1, \ldots, \alpha_r \in \mathbb{R}^n$ of linear combinations of dataset $X \in \mathbb{R}^{n\times m}$ and $\beta_1, \ldots, \beta_r \in \mathbb{R}^p$ of $Z \in \mathbb{R}^{p\times m}$, such that all α_i are orthogonal, all β_i are orthogonal, and $\alpha_i^T X$ and $\beta_j^T Z$ have a maximal correlation with each other for $i = j$ and zero correlation otherwise [11]. Here $r = \min\{\text{rank}(X), \text{rank}(Z)\}$. Thus, CCA seeks vectors α_i and β_j, such that $\rho = \text{cor}(\alpha_i^T X, \beta_j^T Z) = \gamma_i$ is maximal if $i = j$ and $\rho = 0$ otherwise. γ_i are called the canonical correlations. Since all pairs are mutually orthogonal, one has that $\text{cor}(\alpha_i^T X, \alpha_j^T X) = 1$ if $i = j$ and 0 otherwise, and similarly $\text{cor}(\beta_i^T Z, \beta_j^T Z) = 1$ if $i = j$ and 0 otherwise. One will find that α_i is the eigenvector of $\Sigma_X^{-1}\Sigma_{XZ}\Sigma_Z^{-1}\Sigma_{ZX}$ corresponding to the i^{th} largest eigenvalue, and β_i is proportional to $\Sigma_Z^{-1}\Sigma_{ZX}\alpha_i$, where $\Sigma_{XZ} = \text{Cov}(X, Z) = \Sigma_{ZX}^T \in \mathbb{R}^{n\times p}$ are cross-covariance matrices and $\Sigma_X = \Sigma_{XX}$ and $\Sigma_Z = \Sigma_{ZZ}$ are covariance matrices.

Theorem 2.3.10. *The canonical direction $\alpha_i \in \mathbb{R}^n$ of the dataset $X \in \mathbb{R}^{n\times m}$ equals the eigenvector of the matrix $\Sigma_X^{-1}\Sigma_{XZ}\Sigma_Z^{-1}\Sigma_{ZX}$ corresponding to the i^{th} largest eigenvalue, and the canonical direction $\beta_i \in \mathbb{R}^p$ of the dataset $Z \in \mathbb{R}^{p\times m}$ equals $r \cdot \Sigma_Z^{-1}\Sigma_{ZX}\alpha_i$, with $r \in \mathbb{R}$ a constant.*

Proof. Start with the first pair of canonical directions:

$$\begin{aligned}(\alpha_1, \beta_1) &= \underset{\alpha,\beta}{\text{argmax}}\ \text{cor}\left(\alpha^T X, \beta^T Z\right) = \underset{\alpha,\beta}{\text{argmax}}\ \frac{\text{Cov}\left(\alpha^T X, \beta^T Z\right)}{\sigma_{\alpha^T X}\sigma_{\beta^T Z}} \\ &= \underset{\alpha,\beta}{\text{argmax}}\ \frac{\alpha^T \Sigma_{XZ}\beta}{\sqrt{\text{Var}(\alpha^T X)}\sqrt{\text{Var}(\beta^T Z)}} = \underset{\alpha,\beta}{\text{argmax}}\ \frac{\alpha^T \Sigma_{XZ}\beta}{\sqrt{\alpha^T\Sigma_X\alpha}\sqrt{\beta^T\Sigma_Z\beta}}.\end{aligned}$$

Now, change the basis. Substitute with $c = \Sigma_X^{1/2}\alpha \in \mathbb{R}^n$ and $d = \Sigma_Z^{1/2}\beta \in \mathbb{R}^p$ and find:

$$\max_{\alpha,\beta} \frac{\alpha^T \Sigma_{XZ}\beta}{\sqrt{\alpha^T\Sigma_X\alpha}\sqrt{\beta^T\Sigma_Z\beta}} = \max_{c,d} \frac{c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}d}{\sqrt{c^Tc}\sqrt{d^Td}}.$$

Now, use the Cauchy-Schwarz inequality: $(u^Tv)^2 \leq u^Tu \cdot v^Tv$, with $u^T = c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}$ and $v = d$. This gives an upper bound for the numerator

$$\begin{aligned}c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}d &\leq \left(c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}\Sigma_Z^{-1/2}\Sigma_{ZX}\Sigma_X^{-1/2}c\right)^{1/2}\left(d^Td\right)^{1/2}, \\ \max_{c,d}\frac{c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}d}{\sqrt{c^Tc}\sqrt{d^Td}} &\leq \max_c \frac{\left(c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1}\Sigma_{ZX}\Sigma_X^{-1/2}c\right)^{1/2}}{\left(c^Tc\right)^{1/2}},\end{aligned}$$

with equality if d is proportional to c, i.e., $d = r\cdot\Sigma_Z^{-1/2}\Sigma_{ZX}\Sigma_X^{-1/2}c$, with $r \in \mathbb{R}$ a constant $\neq 0$. Substitute this into the inequality:

$$\max_{c,d}\frac{c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}d}{\sqrt{c^Tc}\sqrt{d^Td}} = \max_c \frac{c^T\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}r\cdot\Sigma_Z^{-1/2}\Sigma_{ZX}\Sigma_X^{-1/2}c}{\sqrt{c^Tc}\sqrt{r^2c^T\cdot\Sigma_X^{-1/2}\Sigma_{XZ}\Sigma_Z^{-1/2}\Sigma_Z^{-1/2}\Sigma_{ZX}\Sigma_X^{-1/2}c}}$$

$$\begin{aligned} &= \max_c \frac{c^T \Sigma_X^{-1/2} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1/2} c}{\sqrt{c^T c} \sqrt{c^T \Sigma_X^{-1/2} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1/2} c}} \\ &= \max_c \frac{\left(c^T \Sigma_X^{-1/2} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1/2} c\right)^{1/2}}{\left(c^T c\right)^{1/2}}. \end{aligned}$$

This is essentially the same equation as equation 2.1, and in the proof of Theorem 2.3.5 it was found that this is maximal if c is the eigenvector of $H_c = \Sigma_X^{-1/2} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1/2}$ corresponding to the largest eigenvalue, η. Now, substitute α and β back in:

$$\begin{aligned} H_c c &= \eta c = \eta \Sigma_X^{1/2} \alpha = \Sigma_X^{-1/2} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1/2} \Sigma_X^{1/2} \alpha, \\ \eta \alpha &= \Sigma_X^{-1} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \alpha, \\ d &= \Sigma_Z^{1/2} \beta = r \cdot \Sigma_Z^{-1/2} \Sigma_{ZX} \Sigma_X^{-1/2} c \\ &= r \cdot \Sigma_Z^{-1/2} \Sigma_{ZX} \Sigma_X^{-1/2} \Sigma_X^{1/2} \alpha = r \cdot \Sigma_Z^{-1/2} \Sigma_{ZX} \alpha, \\ \beta &= r \cdot \Sigma_Z^{-1} \Sigma_{ZX} \alpha. \end{aligned}$$

Thus, α_1 is an eigenvector of $H = \Sigma_X^{-1} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX}$ corresponding to the largest eigenvalue η, and β_1 is proportional to $\Sigma_Z^{-1} \Sigma_{ZX} \alpha_1$. Again, similar to the proof of Theorem 2.3.5, one finds by induction that α_j is an eigenvector of $\Sigma_X^{-1/2} H_c \Sigma_X^{1/2} = \Sigma_X^{-1} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} = H$. Given that it must be orthogonal to $\alpha_1, \ldots, \alpha_{j-1}$, it is obtained that α_j must be equal to the eigenvector of $\Sigma_X^{-1} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX}$ that corresponds to the j^{th} largest eigenvalue. Therefore, β_j equals $r \cdot \Sigma_Z^{-1} \Sigma_{ZX} \alpha_j$, with $r \in \mathbb{R}$ a constant. □

α and β can also be switched and $\operatorname{cor}\left(\beta^T Z, \alpha^T X\right)$ can be maximized instead, similar to the proof above it is obtained that β_j must be equal to the eigenvector of $\Sigma_Z^{-1} \Sigma_{ZX} \Sigma_X^{-1} \Sigma_{XZ}$ that corresponds to the j^{th} largest eigenvalue. Therefore, α_j equals $q \cdot \Sigma_X^{-1} \Sigma_{XZ} \beta_j$, with $q \in \mathbb{R}$ a constant. A connection between the constants q and r is found, namely:

$$\begin{aligned} \alpha_j &= q \cdot \Sigma_X^{-1} \Sigma_{XZ} \beta_j, \\ \beta_j &= r \cdot \Sigma_Z^{-1} \Sigma_{ZX} \alpha_j, \\ \alpha_j &= q \cdot r \cdot \Sigma_X^{-1} \Sigma_{XZ} \Sigma_Z^{-1} \Sigma_{ZX} \alpha_j = q \cdot r \cdot H \alpha_j = q \cdot r \cdot \eta \alpha_j, \\ \eta &= \frac{1}{qr}. \end{aligned}$$

For CCA whitening, let X be the input image and Z the set of signatures. Call the whitened matrices $\tilde{X} = W_X X$ and $\tilde{Z} = W_Z Z$. In CCA it is the objective to have $\operatorname{Cov}\left(\tilde{X}, \tilde{Z}\right)$ be a diagonal matrix, with the canonical correlations on its diagonal, i.e.,

$$\begin{aligned} \operatorname{Cov}\left(\tilde{X}, \tilde{Z}\right) &= \mathbb{E}[(W_X X - \mathbb{E}[W_X X])(W_Z Z - \mathbb{E}[W_Z Z])^T] = W_X \mathbb{E}[(X - \mathbb{E}[X])(Z - \mathbb{E}[Z])^T W_Z^T] \\ &= W_X \Sigma_{XZ} W_Z^T = P_{\tilde{X}\tilde{Z}}. \end{aligned}$$

Now, use Theorem 2.3.4 to substitute W_X and W_Z:

$$\begin{aligned} P_{\tilde{X}\tilde{Z}} &= U_X P_X^{-1/2} S_X^{-1/2} \Sigma_{XZ} S_Z^{-1/2} P_Z^{-1/2} U_Z^T \\ &= U_X P_X^{-1/2} P_{XZ} P_Z^{-1/2} U_Z^T = U_X K U_Z^T. \end{aligned}$$

Here, $K = P_X^{-1/2} P_{XZ} P_Z^{-1/2} \in \mathbb{R}^{n \times p}$. Since K is not a square matrix, use a singular value decomposition (SVD) instead of an eigendecomposition. This gives $K = Q_X \Lambda Q_Z^T$, where $Q_X \in \mathbb{R}^{n \times r}$ and $Q_Z \in \mathbb{R}^{p \times r}$ are semi-orthogonal matrices, i.e., $Q_X^T Q_X = Q_Z^T Q_Z = I_r$. Here $r = \min\{n, p\}$. Further, it follows from the SVD that $\Lambda \in \mathbb{R}^{r \times r}$ is a diagonal matrix with the singular values of K on its diagonal. Since U_X and U_Z can be chosen freely, choose them to equal Q_X^T and Q_Z^T respectively. This gives

$$\begin{aligned} P_{\tilde{X}\tilde{Z}} &= U_X K U_Z^T = Q_X^T K Q_Z \\ &= Q_X^T Q_X \Lambda Q_Z^T Q_Z = \Lambda, \end{aligned}$$

which is indeed diagonal. However, unlike in Theorem 2.3.4, U_X and U_Z are not orthogonal, but semi-orthogonal. Nonetheless, it still holds that $U_X U_X^T = U_Z U_Z^T = I_r$, and thus $W \Sigma W^T = I$. Therefore, they can

still be used for whitening [11]. It is found that $W_X = Q_X^T P_X^{-1/2} S_X^{-1/2} \in \mathbb{R}^{r\times n}$ and $W_Z = Q_Z^T P_Z^{-1/2} S_Z^{-1/2} \in \mathbb{R}^{r\times p}$.

Let $A = (\alpha_1,\ldots,\alpha_r)^T \in \mathbb{R}^{r\times n}$ and $B = (\beta_1,\ldots,\beta_r)^T \in \mathbb{R}^{r\times p}$. Now, standardize the canonical correlations, such that $\text{Cov}(\alpha_i^T X, \alpha_i^T X) = \alpha_i^T \Sigma_X \alpha_i = 1$ and $\text{Cov}(\beta_i^T Z, \beta_i^T Z) = \beta_i^T \Sigma_Z \beta_i = 1$. Since $\text{cor}(\alpha_i^T X, \alpha_j^T X) = cor(\beta_i^T Z, \beta_j^T Z) = 0$ if $i \neq j$, it follows that $A\Sigma_X A^T = B\Sigma_Z B^T = I_r$. Thus, A and B are the whitening matrices W_X and W_Z.

2.3.6. Cholesky Whitening

A real positive definite matrix (i.e., all eigenvalues > 0) can be decomposed into the product of the matrices L and L^T, where L is a lower triangular matrix, i.e., $l_{i,j} = 0$ if $i < j$, with positive values on its diagonal. Let $\Sigma^{-1} = LL^T$ be the Cholesky decomposition of Σ^{-1}, the matrix L is unique:

$$\left(\Sigma^{-1}\right)_{i,j} = \left(LL^T\right)_{i,j} = \sum_{k=1}^{n} L_{i,k}\left(L^T\right)_{k,j} = \sum_{k=1}^{n} L_{i,k}L_{j,k},$$

from these equations, all $L_{i,j}$ can beuniquely determined. From $\left(\Sigma^{-1}\right)_{1,1}$ determine $L_{1,1}$: $\left(\Sigma^{-1}\right)_{1,1} = \sum_{k=1}^{n} L_{1,k}L_{1,k} = L_{1,1}^2$. Now, determine $L_{2,1}$ from $\left(\Sigma^{-1}\right)_{1,2}$ and $L_{1,1}$: $\left(\Sigma^{-1}\right)_{1,2} = \sum_{k=1}^{n} L_{1,k}L_{2,k} = L_{1,1}L_{2,1}$. Then, determine $L_{2,2}$ from $\left(\Sigma^{-1}\right)_{2,2}$ and $L_{2,1}$, then $L_{3,1}$ from $\left(\Sigma^{-1}\right)_{1,3}$ and $L_{1,1}$, etc. The following formula's are found:

$$\begin{aligned}
L_{i,i} &= \sqrt{\left(\Sigma^{-1}\right)_{i,i} - \sum_{k=1}^{i-1}\left(L_{i,k}\right)^2} && \text{for } i = j,\\
L_{i,j} &= \tfrac{1}{L_{j,j}}\left(\left(\Sigma^{-1}\right)_{i,j} - \sum_{k=1}^{j-1} L_{i,k}L_{j,k}\right) && \text{for } i > j,\\
L_{i,j} &= 0 && \text{for } i < j.
\end{aligned}$$

Since $\Sigma^{-1} = LL^T$, and the definition of a whitening matrix is that $\Sigma^{-1} = W^T W$, the Cholesky whitening matrix W equals L^T. This corresponds to U_1 from Theorem 2.3.3 being equal to $L^{-1}\Sigma^{-1/2}$, i.e., $W = L^{-1}\Sigma^{-1/2}\Sigma^{-1/2} = L^{-1}\Sigma^{-1} = L^T$. This is indeed orthogonal:

$$\begin{aligned}
U_1^T U_1 &= \Sigma^{-1/2}\left(L^{-1}\right)^T L^{-1}\Sigma^{-1/2} = \Sigma^{-1/2}\left(L^T\right)^{-1} L^{-1}\Sigma^{-1/2} = \Sigma^{-1/2}\left(LL^T\right)^{-1}\Sigma^{-1/2} = \Sigma^{-1/2}\Sigma\Sigma^{-1/2} = I,\\
U_1 U_1^T &= L^{-1}\Sigma^{-1/2}\Sigma^{-1/2}\left(L^{-1}\right)^T = L^{-1}\Sigma^{-1}\left(L^T\right)^{-1} = L^{-1}\left(L^T\Sigma\right)^{-1} = L^{-1}L = I.
\end{aligned}$$

Where it is used that $L^{-1} = L^T\Sigma$. Note that Cholesky decomposition can only be performed if the sample covariance matrix Σ is positive definite, i.e., all eigenvalues should be > 0. Furthermore, a lower triangular matrix is invertible if the elements on the diagonal are not zero. In our case, all diagonal elements are positive, thus L is invertible.

PCA and standardized PCA respectively maximized the cross-covariances ϕ_i and the cross-correlations ψ_i. ZCA and standardized ZCA minimized the Frobenius norm between X and WX, and $S^{-1/2}X$ and WX respectively. Unlike these transformations, Cholesky whitening does not result from an optimization, but from a symmetry constraint. If it is demanded that the cross-covariance matrix Φ and the cross-correlation matrix Ψ are lower triangular matrices with positive elements on the diagonal, then the whitening matrix $W = L^T$ will be found.

Theorem 2.3.11. *If it is demanded that Φ and Ψ are lower triangular matrices with positive diagonal elements, then $W = W_{Chol} = L^T$ must hold, where L is the lower triangular matrix from the Cholesky decomposition of Σ^{-1}.*

Proof. If Φ is lower triangular with positive diagonal elements, then Φ^{-1} exists and it is also lower triangular. This means that there exists a matrix with $\Phi^{-1}(\Phi^{-1})^T$ as its Cholesky decomposition. It will be shown that this matrix equals the inverse of the covariance matrix:

$$\begin{aligned}
\Phi &= U_1\Sigma^{1/2},\\
\Phi^{-1}(\Phi^{-1})^T &= \Sigma^{-1/2}U_1^T U_1\Sigma^{-1/2} = \Sigma^{-1/2}\Sigma^{-1/2} = \Sigma^{-1}.
\end{aligned}$$

The Cholesky decomposition of a matrix is unique, this indicated that $\Sigma^{-1} = LL^T = \Phi^{-1}(\Phi^{-1})^T$, which means that $\Phi^{-1} = L$. Now, an expression for the orthogonal matrix U_1 can be found:

$$U_1 = \Phi\Sigma^{-1/2} = L^{-1}\Sigma^{-1/2}.$$

Similarly, if Ψ is lower triangular with positive diagonal elements, then Ψ^{-1} exists and it is also lower triangular. This means that there exists a matrix with $\Psi^{-1}(\Psi^{-1})^T$ as its Cholesky decomposition. It will be shown that this matrix equals $S^{1/2}\Sigma^{-1}S^{1/2}$:

$$\begin{aligned}\Psi &= U_1\Sigma^{1/2}S^{-1/2},\\ \Psi^{-1}(\Psi^{-1})^T &= S^{1/2}\Sigma^{-1/2}U_1^TU_1\Sigma^{-1/2}S^{1/2} = S^{1/2}\Sigma^{-1}S^{1/2} = S^{1/2}LL^TS^{1/2}.\end{aligned}$$

Since the Cholesky decomposition of a matrix is unique, it follows that $\Psi^{-1} = S^{1/2}L$. The same expression is found for U_1:

$$\begin{aligned}\Psi &= L^{-1}S^{-1/2} = U_1\Sigma^{1/2}S^{-1/2},\\ U_1 &= L^{-1}\Sigma^{-1/2}.\end{aligned}$$

From this expression of U_1, it indeed follows that $W_{Chol} = U_1\Sigma^{-1/2} = U_1\Sigma^{-1/2}\Sigma^{-1/2} = L^{-1}\Sigma^{-1} = L^{-1}LL^T = L^T$. □

2.3.7. Characteristics of Whitening Transformations

A small overview of the whitening matrices that were discussed is given: what the whitening matrix W looks like and how it is found. The other characteristics can be found in Table 2.1.

- PCA maximizes $\phi_i = \sum_{j=1}^n \phi_{i,j}^2 = \sum_{j=1}^n \mathrm{Cov}(WX_i, X_j)^2$, giving $W_{PCA} = D^{-1/2}V^T$.
- Standardized PCA maximizes $\psi_i = \sum_{j=1}^n \psi_{i,j}^2 = \sum_{j=1}^n \mathrm{cor}(WX_i, X_j)^2$, giving $W_{PCA-std} = \Theta^{-1/2}G^TS^{-1/2}$.
- ZCA minimizes $||X - WX||_F^2$, and is therefore closest to the original data, giving $W_{ZCA} = \Sigma^{-1/2}$
- Standardizes ZCA minimizes $||S^{-1/2}X - WX||_F^2$, and is therefore closest to the standardized data, giving $W_{ZCA-std} = P^{-1/2}S^{-1/2}$.
- CCA finds canonincal directions $\alpha_1, \ldots, \alpha_r \in \mathbb{R}^n$ of linear combinations of dataset $X \in \mathbb{R}^{n\times m}$ and $\beta_1, \ldots, \beta_r \in \mathbb{R}^p$ of $Z \in \mathbb{R}^{p\times m}$, such that all α_i are orthogonal, all β_i are orthogonal, and α_i^TX and β_j^TZ have a maximal correlation with each other for $i = j$ and zero correlation otherwise. This gives $W_X = Q_X^TP_X^{-1/2}S_X^{-1/2}$ and $W_Z = Q_Z^TP_Z^{-1/2}S_Z^{-1/2}$, where $K = Q_X\Lambda Q_Z^T$ is the singular value decomposition (SVD) of $K = P_X^{-1/2}P_{XZ}P_Z^{-1/2}$.
- Cholesky whitening does not result from an optimization, unlike these other transformations, but from a symmetry constraint. It is demanded that the cross-covariance matrix Φ and the cross-correlation matrix Ψ are lower triangular matrices with positive elements on the diagonal, giving $W_{Chol} = L^T$, where $\Sigma^{-1} = LL^T$.

	Whitening Matrix W	Rotation Matrix $U_1 = W\Sigma^{1/2}$	Rotation Matrix $U_2 = U_1R^T$	Cross-covariance $\Phi = U_1\Sigma^{1/2}$	Cross-correlation $\Psi = U_2P^{1/2} = \Phi S^{-1/2}$
PCA	$D^{-1/2}V^T$	V^T	V^TR^T	$D^{1/2}V^T$	$D^{1/2}V^TS^{-1/2}$
PCA-std	$\Theta^{-1/2}G^TS^{-1/2}$	G^TR	G^T	$G^TP^{1/2}S^{1/2}$	$\Theta^{1/2}G^T$
ZCA	$\Sigma^{-1/2}$	I	R^T	$\Sigma^{1/2}$	$\Sigma^{1/2}S^{-1/2}$
ZCA-std	$P^{-1/2}S^{-1/2}$	R	I	$P^{1/2}S^{1/2}$	$P^{1/2}$
Cholesky	L^T	$L^T\Sigma^{1/2}$	$L^TS^{1/2}P^{1/2}$	$L^T\Sigma$	$L^T\Sigma S^{-1/2}$
CCA	$W_X = Q_X^TP_X^{-1/2}S_X^{-1/2}$	$U_1^X = Q_X^TR_X$	$U_2^X = Q_X^T$	$\Phi_X = Q_X^TP_X^{1/2}S_X^{1/2}$	$\Psi_X = Q_X^TP_X^{1/2}$

Table 2.1: Different whitening transformations and their characteristics.

3

Methodology

In this Chapter, the methods used to create the results given in Chapter 4 are discussed. First of all, synthetic data is created to analyse the effects of whitening without any unknown influences from potential undocumented artefacts in real data. Creating synthetic data also means a much smaller amount of pixels can be studied than the amount of pixels in a full scan, such that the effects of whitening can be seen more clearly. So, the first section will be a step-by-step description of how the synthetic data is made. This is followed by a description of the empirical whitening procedure, including what measures were taken when the empirical covariance matrix was not positive definite, and including an explanation to which whitening transformations were used and which were not, and why. Next, the step-by-step process is given of how the Signal-to-Noise Ratio (SNR) was calculated. Subsequently, a description is given of how the Monte Carlo simulations were done, to give a better view of the signal-to-noise ratio. And lastly, a description is given of the methods used for the multi-area whitening procedure, which consists of the creation of another synthetic spectrum, and testing how it influences the target spectrum in a high contrast scenario. This will be measured using the cosine similarity. The code for all procedures described in this chapter can be found in Appendix C.

3.1. Synthetic Data

First, a substrate must be chosen to display the target substance on. Stage Gate 11 B.V. uses polytetrafluorethylene (PTFE) as a substrate, because it possesses certain photometric qualities. PTFE is also known under the brand name Teflon, Stage Gate 11 B.V. specifically uses white Teflon. They put a known amount of target substance onto the Teflon before it goes through the scanner. Thus, in order to simulate this process and create a synthetic scan, a background pixel is created first, i.e., a Teflon spectrum. Then, a spectrum for a target pixel is created, specifically for when a target substance is placed upon a piece of white Teflon. After that, noise is added to the created spectra, because in reality, the data is subject to shot noise. At this point, two spectra have been made: one of a piece of white Teflon and the other of a piece of white Teflon with a target substance on it. Now, a synthetic scan can be created from these spectra with any number of pixels in it. The underlying text explains how the synthetic data was created, step by step.

1. Creating a background pixel
   First, a scan of pure Teflon is selected. From this scan, 50 pixels were chosen, such that the selected pixels were non-adjacent. This is done to limit the effect of possible irregularities, e.g., a bit of dust on the Teflon piece. Then, for every wavelength the average intensity over all 50 Teflon pixels is calculated, creating an average Teflon spectrum. This average spectrum still contains noise, so a polynomial is fitted to the data to create a smooth spectrum without noise. A polynomial with degree 6 was found to be adequate, as can be seen in Figure 3.1. The fitted polynomial is of the form:

   $$z_0 + z_1 \cdot x + z_2 \cdot x^2 + z_3 \cdot x^3 + z_4 \cdot x^4 + z_5 \cdot x^5 + z_6 \cdot x^6,$$

   where

- $z_0 = 2.08134265 \cdot 10^6$,
- $z_1 = -3.95708439 \cdot 10^4$,
- $z_2 = 3.06589766 \cdot 10^2$,
- $z_3 = -1.23925005$,
- $z_4 = 2.75658174 \cdot 10^{-3}$,
- $z_5 = 1.49316921 \cdot 10^{-9}$,
- $z_6 = 1.49316921 \cdot 10^{-9}$.

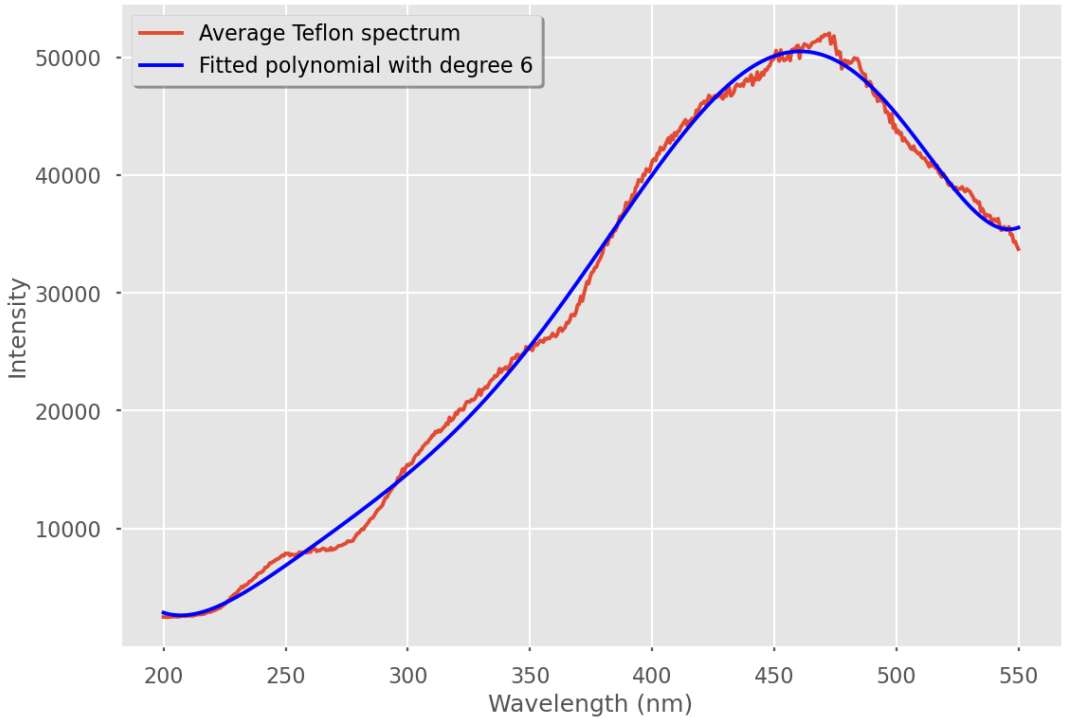


Figure 3.1: Spectrum of the average of 50 Teflon pixels and the polynomial with degree 6 fitted to that average Teflon spectrum.

2. Creating a target pixel
   In reality, a target pixel will look similar to the average Teflon spectrum, but with a Gaussian distribution subtracted from it. This is because the situation that was chosen to be simulated is the situation where the target substance to be detected is placed upon a piece of Teflon. Thus, the spectrum will look like the Teflon spectrum, except at the wavelengths where the target substance absorbs the light. Therefore, the location of this Gaussian depends on the chosen substance. The Gaussian itself is representative of a signature from an explosive [21]. The amplitude depends on the amount of substance, a higher amount means more absorption and thus a bigger amplitude. The standard deviation also depends on the substance. For our target pixel, a shift of 300 and standard deviation of 10 are chosen, i.e., the Gaussian will have $\mu = 300$, $\sigma = 10$. Furthermore, an amplitude A of $\frac{150.000}{\sqrt{2\pi}}$ is chosen. This results in the following probability density function:

$$f(x) = \frac{A}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma^2}\right)^2} = \frac{150.000}{20\pi} e^{-\frac{1}{2}\frac{(x-300)^2}{100}}. \tag{3.1}$$

   The resulting Gaussian probability density can be found in Figure 3.2.

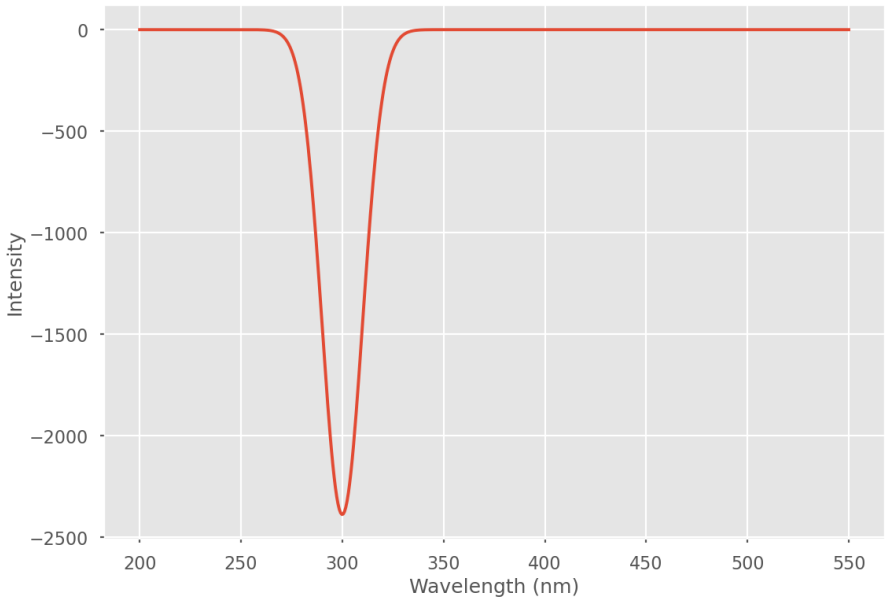


Figure 3.2: Negative Gaussian probability density function, $-f(x)$, as made in Equation 3.1.

Subtracting the Gaussian probability density function as in Equation 3.1 from the average Teflon spectrum, i.e., adding the negative Gaussian distribution as in Figure 3.2, results in spectrum of the target pixel, depicted in Figure 3.3.

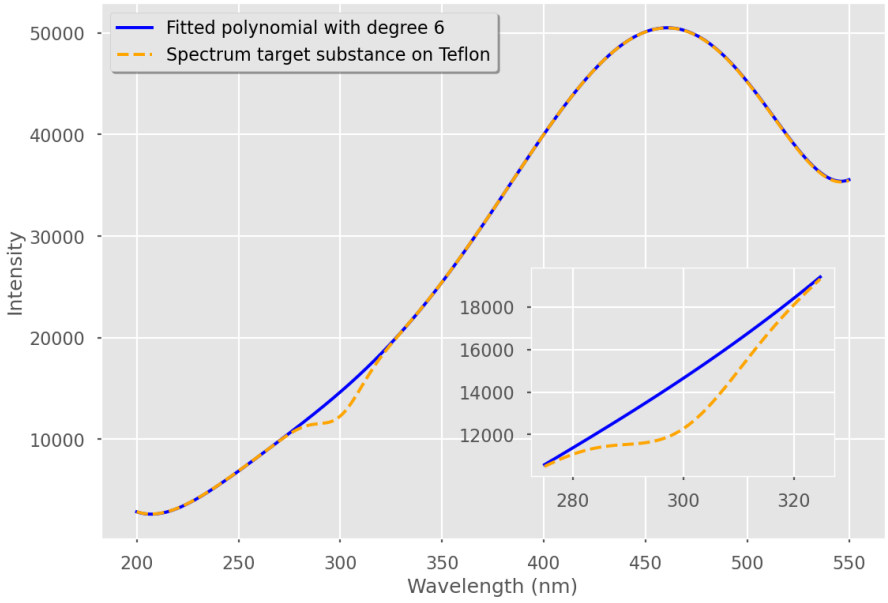


Figure 3.3: The polynomial with degree 6 fitted to the average Teflon spectrum and the target pixel made from the Teflon polynomial, enlarged between 275 and 325 nm.

3. Adding noise
   The solutions produced by Stage Gate 11 B.V. use a spectrograph with an imaging camera based upon a scientific Complementary Metal Oxide Semiconductor (sCMOS) imaging chip. Such a chip produces certain types of noise. A sCMOS device is subject to three types of noise: shot noise, dark noise and read-out noise [10, 23]. Due to this notion, noise needs to be added to synthetic data to mimic the actual data as accurately as possible. Of the three types of noise, shot noise is the predominant component

of noise, being typically bigger by at least an order of magnitude compared to dark noise and read noise. Thus, only shot noise is considered to be required to be added as other components will have a comparatively negligible effect. Thus, shot noise is added to all pixels. The shot noise has a Gaussian distribution with $\mu = 0$, $\sigma^2 = \sqrt{I}$, i.e., the variance is proportional to the square root of the intensity I [10, 23]. A higher intensity gives a larger shot noise.

4. Creating a synthetic scan
   Lastly, a synthetic scan can be created. If a scan is of size n, then $n-1$ Teflon pixels and 1 target pixel are created. Half of the Teflon pixels are placed in an array, then the target pixel is added and finally the rest of the Teflon pixels. This means that a matrix of size $n \times 548$ is immediately created, instead of creating a three- dimensional data cube first. This two-dimensional array is the synthetic scan. If n is odd, first $\frac{n-1}{2}$ Teflon pixels are added to the array, then the target pixel and then another $\frac{n-1}{2}$ Teflon pixels, meaning that the target pixel is pixel $\frac{n+1}{2}$, which is exactly in the middle. If n is even, $\frac{n}{2}$ Teflon pixels are added to the array, then the target pixel and then another $\frac{n}{2}-1$ Teflon pixels, meaning that the target pixel is pixel $\frac{n}{2}+1$, which is not exactly in the middle, as that is not possible with an even number of pixels. When all pixels are created for the scan, it is made sure that different random numbers are generated for each pixel when simulating the shot noise.

3.2. Whitening

3.2.1. Empirical Whitening Process

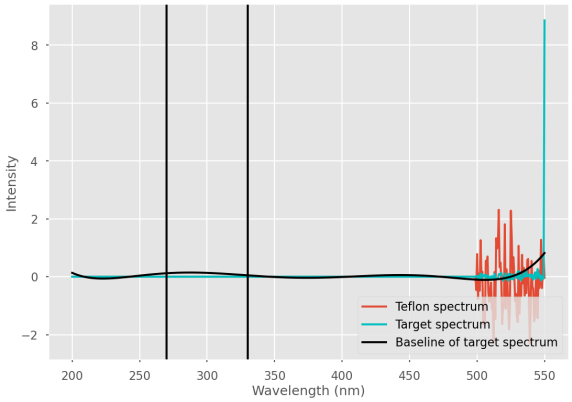
When starting the whitening process, the data is first centered, such that the sum of each row equals 0. Then, the empirical covariance matrix is calculated. The empirical covariance matrix $\hat{\Sigma} = \frac{1}{m-1} XX^T$, will be of size $n \times n$. This means that if pixels are used as observations, $\hat{\Sigma}$ will always be of size 548×548. Therefore, the covariance matrix can never be positive definite if there are less than 548 pixels in a scan. There will be eigenvalues equal to zero, meaning that the empirical covariance matrix cannot be inverted. Therefore, an epsilon is added to the eigenvalues in D. If wavelengths are used as observations, $\hat{\Sigma}$ will always be of size $n \times n$, where n is the number of pixels, which varies per scan. In this case, the covariance matrix can never be positive definite if there are more than 548 pixels in a scan. Again, there will be eigenvalues equal to zero, so an epsilon needs to be added to the eigenvalues in D. In all cases, $\epsilon = 1e^{-4}$ is used, even if the matrix is positive definite, for simplicity. The empirically acquired matrices V and D are then used to calculate a whitening matrix W, depending on which whitening transformation is chosen. Finally, W is multiplied with the centered dataset X, this gives the whitened image $Y = WX$.

3.2.2. Used Whitening Transformations

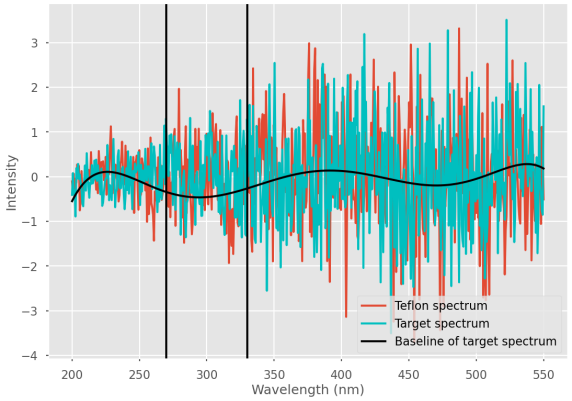
Unfortunately, it turns out that not all discussed whitening transformations in Section 2.3 are useful for detecting illicit substances. In this section, a discussion on what transformations were and were not used will be given and an explanation why.

3.2.2.1. PCA and Standardized PCA Whitening

The PCA whitening matrix is determined using the eigendecomposition $\Sigma = VDV^T$, such that $W_{PCA} = D^{-1/2}V^T$. This means that the data is first rotated by V^T and then scaled by $D^{-1/2}$. However, the eigenvalues in D are ordered from smallest to largest. When pixels are used as observations, the big peak is not at 300 nm anymore, instead it will always be at 550 nm, which is not correct. This can be fixed by rotating the data back using V, but that will give $W = VD^{-1/2}V^T$, which is just ZCA whitening. Furthermore, if a 9×9 synthetic scan is made, it will contain 81 pixels. This means that there are 81 eigenvalues bigger than or equal to zero, and 548-81 = 467 eigenvalues that are definitely zero. Because there is no rotation back to the original coordinate system, this can be seen in the whitened spectrum, see Figure 3.4a. When wavelengths are used as observations, there will be whitened data that is not equal to zero, but no clear signal can be made out, as seen in Figure 3.4b. Therefore, it is decided that PCA whitening will not be used for the rest of this report. Standardized PCA has the same problems, so it is also not used.



(a) Pixels used as observations on a 9 × 9 scan.



(b) Wavelengths used observations on a 2 × 2 scan.

Figure 3.4: PCA whitening of two synthetic scans using pixels (3.4a) and wavelengths (3.4b) as observations.

3.2.2.2. ZCA and Standardized ZCA Whitening

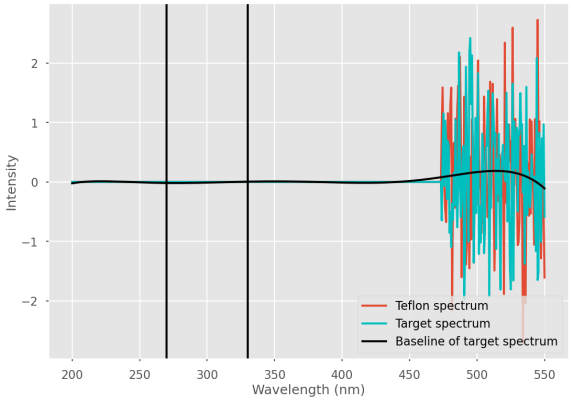
ZCA whitening will be used, so the results of that will be discussed in the next chapter, Chapter 4. As for standardized ZCA, the available time did not permit the completion of the code of this whitening transformation. The code can still be found in Appendix C.

3.2.2.3. CCA Whitening

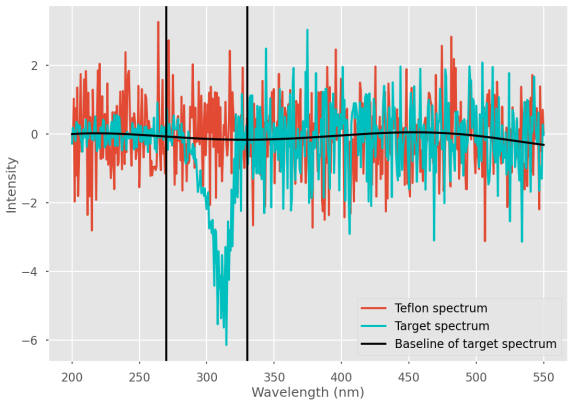
For CCA whitening, two sets need to be whitened simultaneously. The set X, the input image, i.e., the row-wise centered data matrix, and Z, the set of signatures. As there was chosen to work with synthetic data, there exists no set of signatures Z. Also, this method works quite differently from the other methods and due to time constraints, the coding of this whitening transformation was not completed. The code can still be found in Appendix C.

3.2.2.4. Cholesky Whitening

When pixels are used as observation, the empirical covariance matrix will not be positive definite if the number of pixels in the scan is smaller than 548. This means a Cholesky decomposition cannot be performed. However, adding the epsilon, $\epsilon = 1e^{-4}$, will help run the code. Nevertheless, the same problems as for PCA are still there, where all the data is zero or very close to zero for the number of pixels that 'lack' to reach 548, i.e., if a 11 × 11 synthetic scan is made, it will contain 121 pixels. This means that there are 121 eigenvalues bigger than or equal to zero, and 548-121 = 427 eigenvalues that are definitely zero, resulting in the first 427 datapoints having little to no signal, see Figure 3.5a. The number of pixels needs to be bigger than 548, only then a peak around 300 nm starts to show, see Figure 3.5b. However, this peak seems to be skewed and does not lie perfectly around 300 nm anymore.



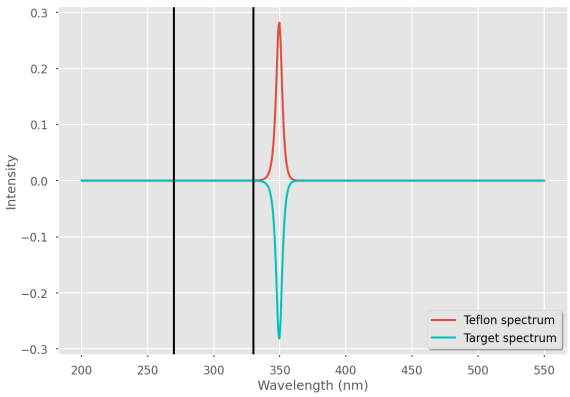
(a) Pixels used as observations on a 11 × 11 scan.



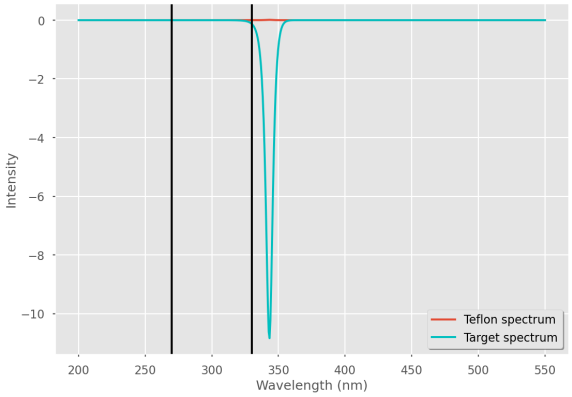
(b) Pixels used observations on a 29 × 29 scan.

Figure 3.5: Cholesky whitening of two different sized synthetic scans.

Furthermore, when whitening a scan without shot noise added, the peak of the Gaussian does not lie at the correct location, if pixels are used as observations. In the cases that were researched, which is scans up to and including a size of 1000 pixels, it did not matter if the scan contains more or less than 548 pixels, see Figure 3.6. Therefore, it is chosen to not work with Cholesky whitening.



(a) Two spectra of a 1×2 synthetic scan, i.e., a scan consisting of one Teflon pixel and one target pixel, after Cholesky whitening, pixels used as observations.



(b) Two spectra of a synthetic scan with 841 pixels in total after Cholesky whitening, the target spectrum and one of the Teflon spectra, pixels used as observations.

Figure 3.6: Two spectra after Cholesky whitening, the target spectrum and the Teflon spectrum, for different sizes of synthetic scans, pixels used as observations.

In conclusion, the results will focus solely on ZCA whitening.

3.3. Signal-to-Noise Ratio

In order to identify a target substance on a substrate, it is important that a signal can be made out beyond the noise, if not, then it cannot be classified. To quantify the relationship between the desired signal and the unwanted noise, the Signal-to-Noise Ratio (SNR) can be calculated. The signal-to-noise ratio is the essential figure-of-merit in assessing performance of the whitening transformation [3]. A higher signal-to-noise ratio means a higher ability to detect meaningful patterns within the whitened spectra. As mentioned in the Introduction, Chapter 1, there is reason to believe that there exists a connection between the amount of whitened pixels in the scene and the signal-to-noise ratio. Therefore, the signal-to-noise ratio will be calculated for different sizes of synthetic scans. This can help with finding suitable cut-off criteria for multi-area whitening. A description of how the signal-to-noise ratio is calculated will follow.

1. Fitting the baseline
   When pixels are used as observations, there is no clear visible baseline underneath the whitened data. However, this baseline can be clearly observed when using wavelengths as observations, specifically when using a small amount of pixels, as in Figure 3.7. To correctly calculate the SNR, this baseline must be calculated and subtracted from the data. Therefore, a polynomial is fitted to the data without the Gaussian. The Gaussian is centered around 300 with $\sigma = 10$, so the data from wavelengths $300 \pm 3\sigma$ is not used for this fitting, i.e., from 270 nm to 330 nm, see the vertical lines in Figure 3.7. Later, the integral of the Gaussian has to be calculated, so then the data from 270 nm to 330 nm is used. To avoid correlation, 10 datapoints away from 300 nm are taken extra. The spectrum goes from 200 nm to 550 in 548 steps, so x[110] = 270,384 and x[203] = 329,89. Taking 10 extra datapoints away from 300 nm means the data from x[0] up to and including x[100] and from x[213] up to and including x[547] will be used. Since the average Teflon spectrum that was used was a polynomial with degree 6, again a polynomial with a degree of 6 is used to fit the baseline to this data.

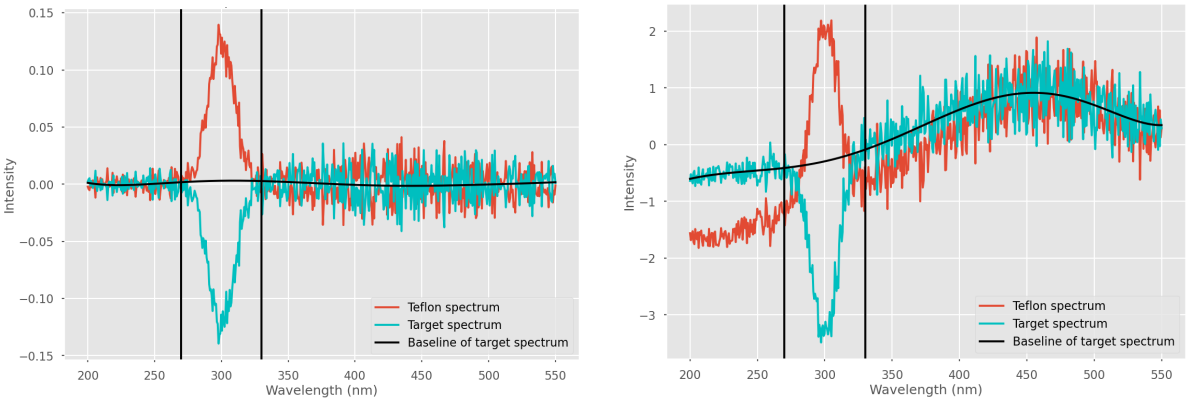
2. Calculating the noise, i.e., σ_{RMS}
   To calculate the noise, first the baseline is subtracted from the data. Then, the standard deviation of the same data where the baseline was fitted on is calculated, i.e., the data from x[0] up to and including x[100] and from x[213] up to and including x[547].

3. Calculating the signal, i.e., the integral of the Gaussian
   Again, the baseline is subtracted from the data. Then, the integral of the Gaussian must be calculated,

or in our case, estimated. The aim is to integrate between 270 nm and 330 nm, but this time, starting from 10 datapoints extra toward the 300 nm to avoid correlation. The spectrum goes from 200 nm to 550 in 548 steps, so x[110] = 270,384 and x[203] = 329,89. Taking 10 extra datapoints toward from 300 nm means that the data used lies between x[120:194]. Then, the integral I is approximated with a midpoint Riemann sum [14].

4. Calculating the Signal-to-Noise Ratio
   The signal-to-noise ratio is the ratio between the desired signal and the background noise. In our case, the desired signal is the integral I around 300 nm, and the background noise is standard deviation σ_{RMS} of the shot noise outside the signal. This gives the formula

$$SNR = \frac{I}{\sigma_{RMS}}.$$



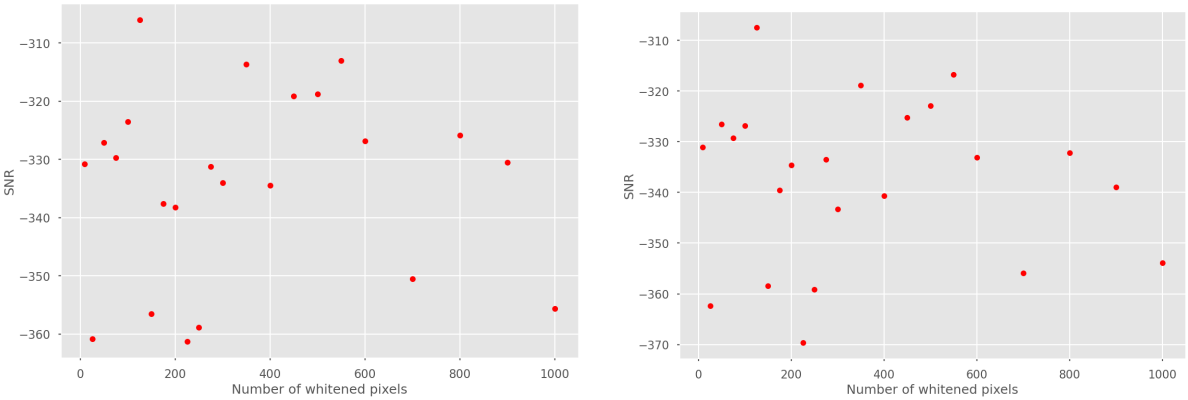
(a) Pixels used as observations.

(b) Wavelengths used observations.

Figure 3.7: ZCA whitening of synthetic scan with two pixels: one Teflon pixel and one target pixel. And fitted baseline of target pixel using pixels (3.7a) and wavelengths (3.7b) as observations.

3.4. The Influence of the Number of Pixels in the Scene

It is suspected that the number of pixels in the scene, i.e., the number of pixels in the input image, influences the signal-to-noise ratio. Therefore, the signal-to-noise ratio is calculated, using the steps described in Section 3.3, for different amounts of pixels in the scan. Synthetic scans of sizes 9, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 350, 400, 450, 500, 550, 600, 700, 800, 900 and 1000 are created and the signal-to-noise ratio of the target pixel is calculated. The results can be found in Figure 3.8.



(a) Pixels used as observations.

(b) Wavelengths used observations.

Figure 3.8: Signal-to-noise ratios for ZCA and whitening against the number of pixels in the synthetic scan, with pixels and wavelengths used as observations.

It is difficult to see any trends in this data. Since noise is added to the spectral data inside each pixel, it is possible that this added noise creates variability of the results beyond the magnitude of the trend one wishes to observe. Therefore, Monte Carlo simulations are used to reveal the patterns which otherwise remain obscured.

3.4.1. Monte Carlo Simulations

Monte Carlo methods are found on the connection between volume and probability. They build upon the mathematical concept of measure. This mathematical framework associates an event with a specific set of potential outcomes and defines the event's probability as the ratio between the volume or measure of the event and that of all potential outcomes. Monte Carlo simulations reverse this identity, determining the set's volume by interpreting it as a probability. This approach involves randomly sampling from all potential outcomes and calculating the fraction of these random draws that fall within a specific set. This fraction serves as an estimate of the set's volume. As the number of random draws increases, the law of large numbers guarantees that this estimate will converge towards the true volume of the set. Additionally, the central limit theorem provides valuable insights into the likely magnitude of error in the estimate after a finite number of draws M [8].

In our case, Monte Carlo simulations are used to estimate the expected signal-to-noise ratio. As the data is subject to noise, trends are difficult to observe. Using M Monte Carlo simulations and taking the average signal-to-noise ratio, gives an estimate of the expected value. The law of large numbers guarantees that this estimate will converge towards the true expected value of the signal-to-noise ratio as M increases. $M = 1000$ is deemed to be sufficiently large.

As described above, 23 synthetic scans of various sizes are created, the signal-to-noise ratio is calculated and saved in a matrix. This process is repeated 1000 times, thus creating a matrix of size 23×1000. Then, for all 23 rows, the average and the standard deviation are calculated. This results in two arrays of length 23, where the 23 entries correspond to the number of whitened pixels in the scan, namely: 9, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 350, 400, 450, 500, 550, 600, 700, 800, 900 and 1000. The first array contains the average signal-to-noise ratio of 1000 Monte Carlo simulations as its entries. The second array contains the standard deviation of these 1000 Monte Carlo simulations as its entries.

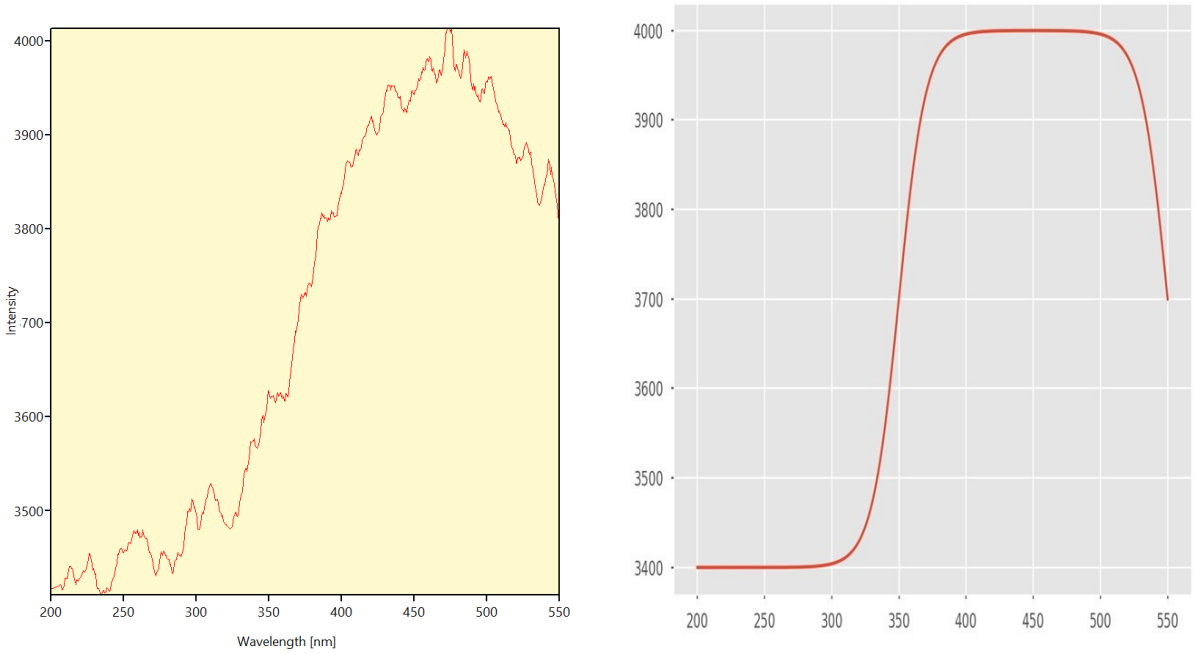
3.5. Multi-area whitening

When an object to be scanned contains more than one background material, the resulting spectra after whitening of all materials can be influenced by the intermixing of the various input spectra before whitening. To prevent this intermixing, a different method can be applied: multi-area whitening. In multi-area whitening, the complete scene is divided into separate categories of the individual substrates. These categories are individually whitened. After whitening, all whitened pixels are combined to create the whitened image. By doing this, the influence of the other materials on the spectra that one could get if all pixels are whitened at the same time, is avoided.

In this research, the choice was made to work with two different materials. The first material is Teflon, for which a synthetic spectrum was created earlier. For the second material, a new synthetic spectrum must be created. It was chosen to recreate the spectrum of leather.

3.5.1. Creating the Leather Spectrum

To create a leather spectrum, a real recorded spectrum of a leather pixel was selected for recreation, as seen in Figure 3.9a. It was recreated by eye, and a logistic curve was deemed to be of accurate shape. Two curves were used, as the intensity of the spectrum increases and then decreases. The increasing part is roughly from $x = 0$ up to $x = 450$, and the decreasing part is roughly from $x = 450$ up to and including $x = 550$.



(a) An example of a real spectrum

(b) Synthetic spectrum

Figure 3.9: Real and synthetic spectrum of a leather pixel.

The logistic curve follows the formula

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$

where

- x_0 is the x value of the midpoint of the function, i.e., the point where the second derivative equals zero $f''(x_0) = 0$,
- L is the supremum of the function,
- k is the logistic growth rate, i.e., how steep the logistic curve is.

For the first function $x_0 = 350$ was chosen, and for the second function $x_0 = 550$ was chosen to be the midpoint of the function. The leather spectrum starts around an intensity of 3400, so that number is added to the first function. It then increases to around 4000, so $L = 600$ is used. The second (decreasing) function starts at 4000 and then decreases back to 3400, so 4000 is added and $L = 600$ is used, but the logistic curve is now multiplied by -1. And lastly, $k = \frac{1}{10}$ is selected for the steepness of the curve. The following two functions follow from this:
Let x be the wavelength,

1. if $x < 450$, then $f(x) = 3400 + \frac{600}{1+(e^{-1/10}\cdot(x-350))}$,
2. if $x \geq 450$, then $f(x) = 4000 - \frac{600}{1+(e^{-1/10}\cdot(x-550))}$.

Here, $f(x)$ is the intensity at wavelength x. The synthetic spectrum as seen in Figure 3.9b is created.

However, as seen in Figure 3.1, the Teflon spectrum reaches an intensity up to 50.000. Therefore, it has been decided to increase the amplitude of the leather spectrum, such that it has a similar intensity. This may not be realistic, but in this way, better research on the influence of other materials in a scan on the whitening transformation can be done. Thus, the following formulas are used:
Let x be the wavelength,

1. if $x < 450$, then $f(x) = 3400 + \frac{46600}{1+(e^{-1/10}\cdot(x-350))}$,
2. if $x \geq 450$, then $f(x) = 50000 - \frac{46600}{1+(e^{-1/10}\cdot(x-550))}$.

This gives a spectrum with the same shape as in Figure 3.9b, but now ranging in intensity from 3400 to 50.000.

3.5.2. The Influence of Shifting the Spectrum on the Scenario

In order to test the effect of multi-area whitening versus global whitening, it is desirable to have as much influence on the spectrum of the target pixel, caused by the leather pixel, as possible. Therefore, the leather spectrum is shifted to the left and the right and to determine how the target spectrum was affected.

First, the scenario used to test this influence must be discussed. A 6×7 simulated scan was created, where the first three rows contain Teflon pixels, the last three rows contain leather pixels. The pixel in the middle of the Teflon pixels is a target pixel, see Figure 3.10.

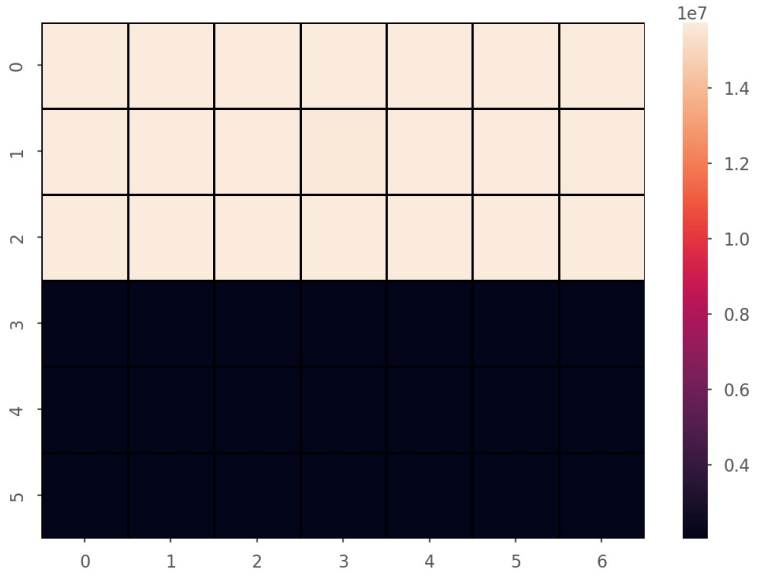


Figure 3.10: Heatmap of the multi-area whitening scenario. Rows 0, 1 and 2 contain Teflon pixels. Rows 3, 4 and 5 contain leather pixels. Row 1 column 3 contains the target pixel on a Teflon background.

Then, the leather spectrum was shifted to the left and the right, to test when the biggest influence on the target spectrum was obtained. This process uses the following formulas:
Let x be the wavelength,

1. if $x < n$, then $f(x) = 3400 + \frac{46600}{1+(e^{-1/10} \cdot (x-(n-100)))}$,

2. if $x \geq n$, then $f(x) = 50000 - \frac{46600}{1+(e^{-1/10} \cdot (x-(n+100)))}$.

For n, the numbers 407.95, 423.95, 439.95, 455.94, 471.94, 487.93, 503.93 and 519.93 were used. These numbers correspond to x[325], x[350], x[375], x[400], x[425], x[450], x[475] and x[500] respectively. Note that 450 gives the original spectrum, this is roughly x[391] = 450.18, thus n = 400 lies closest to the original leather spectrum. The unwhitened spectra can be found in Appendix B in Figures B.1, B.3, B.5, B.7, B.9, B.11, B.13 and B.15, respectively. The results after ZCA whitening the entire scan at once, i.e., global ZCA whitening, and whitening the two categories separate, i.e., multi-area ZCA whitening, can be found in Appendix B in Figures B.2, B.4, B.6, B.8, B.10, B.12, B.14 and B.16, respectively.

In these figures, it can be seen that the shape of the spectrum of the target pixel is highly affected by the presence of leather in the scene. Although the shape of the spectra around 300 nm are quite similar, after the 300 nm, the spectrum of a target pixel where global whitening was used, deviates from the spectrum of a target pixel where multi-area whitening was used. Because of this change in shape, the baseline fit does not work properly anymore, see Figure 3.11

Figure 3.11: Spectra of the target pixel after using global ZCA whitening and multi-area ZCA whitening, and the baseline that is fitted to the target spectrum after global ZCA whitening, pixels used as observations.

Therefore, the signal-to-noise ratio cannot be properly calculated using the steps described in Section 3.3. But as Stage Gate 11 B.V. mostly uses the shape of the spectrum to determine the substance, it will be interesting to investigate the shape instead of the signal-to-noise ratio. Thus, the similarity between the target spectrum using global whitening and the target spectrum using multi-area whitening must be determined. To do this, the spectral angle or cosine similarity between the spectra will be used.

3.5.3. Spectral Angle and Cosine Similarity

In this section, the spectral angle and cosine similarity will be discussed, these are both commonly used for comparing hyperspectral images. The spectral angle mapper (SAM) is a classification algorithm that calculates the spectral angle θ between a pixel P in the image and a reference spectrum Q. SAM is relatively robust against variation in the total illumination intensity, as it exclusively compares the vectors direction on a band-wise basis, such that the length of the vector does not affect the final spectral angle [26].

$$\theta = \arccos\left(\frac{\sum_{i=1}^{548} p_i q_i}{\left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{548} q_i^2\right)^{\frac{1}{2}}}\right).$$

The cosine of the angle θ is called the cosine similarity.

$$S_C(P,Q) = \cos(\theta) = \frac{\sum_{i=1}^{548} p_i q_i}{\left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{548} q_i^2\right)^{\frac{1}{2}}}.$$

As the spectral angle θ ranges from 0 to 180°, the cosine similarity ranges from 1 to -1, see Figure 3.12.

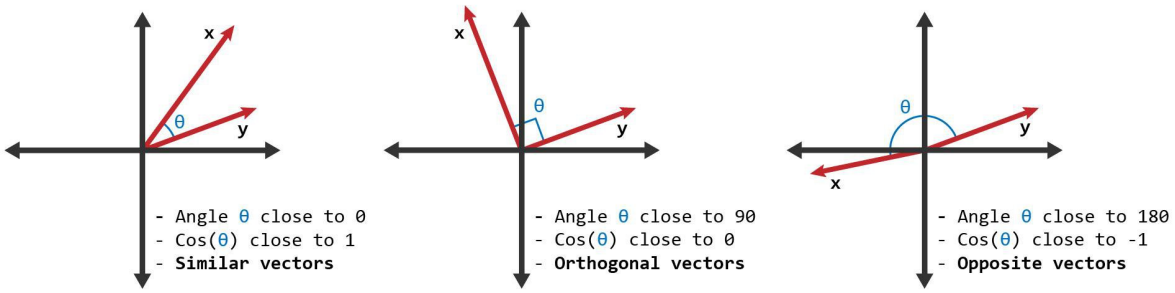


Figure 3.12: Different spectral angles between two vectors and their corresponding cosine similarities, figure taken from [12].

Figure 3.12 contains three statements about vectors, that connect the cosine similarity of two spectra P and Q. Note that $-1 \leq S_C(P,Q) \leq 1$.

1. The cosine similarity equals 1 if and only if the spectra are exactly the same, i.e., $P = Q$:

$$S_C(P,P) = \frac{\sum_{i=1}^{548} p_i^2}{\left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}}} = \frac{\sum_{i=1}^{548} p_i^2}{\sum_{i=1}^{548} p_i^2} = 1.$$

2. The cosine similarity equals -1 if and only if the spectra are exactly each other opposites, i.e., $P = -Q$:

$$S_C(P,-P) = \frac{\sum_{i=1}^{548} -p_i^2}{\left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{548} (-p_i)^2\right)^{\frac{1}{2}}} = \frac{-\sum_{i=1}^{548} p_i^2}{\sum_{i=1}^{548} p_i^2} = -1.$$

3. The cosine similarity will equal 0 if and only if the spectra are orthogonal, i.e., $\sum_{i=1}^{548} p_i q_i = 0$, so clearly:

$$S_C(P,Q) = \frac{\sum_{i=1}^{548} p_i q_i}{\left(\sum_{i=1}^{548} p_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{548} q_i^2\right)^{\frac{1}{2}}} = 0.$$

Taking the arccos of the cosine similarity then gives a value $\theta \in [0,\pi)$ when using radians, where a smaller angle means a higher similarity. As the cosine similarity needs to be calculated in order to calculate the spectral angle, the cosine similarity is selected for comparing spectra. They are both perfectly understandable, but this saves us the computation of an arccos.

Now, the cosine similarity can be calculated for different shifts of the spectrum, to determine for which shift the target spectrum using global whitening and the target spectrum using multi-area whitening are the least similar. The cosine similarity between the target spectrum after global ZCA whitening and the target spectrum after multi-area ZCA whitening is calculated, thus indicating for which shift the target spectrum changes the most. The results can be found in Table 3.1, here pixels were used as observations.

i	325	350	375	400	425	450	475	500
$n = x[i]$	407.95	423.95	439.95	455.94	471.94	487.93	503.93	519.93
Cosine Similarity, pixels as observations	0.9954	0.9722	0.8996	0.8638	0.8848	0.9196	0.9367	0.9594
Cosine Similarity, wavelengths as observations	0.9961	0.9696	0.9073	0.8857	0.9430	0.9672	0.9828	0.9913

Table 3.1: Cosine similarity between the target spectrum after global ZCA whitening and the target spectrum after multi-area ZCA whitening, for different shifts in wavelength n of the leather spectrum in the scenario described in Section 3.5.2, pixels and wavelengths used as observations.

It can be seen that a shift of $i = 400$ gives the lowest cosine similarity, for both pixels and wavelengths used as observations. Furthermore, a shift of $i = 400$, i.e., $n \approx 455.94$ is the closest to the actual leather spectrum, where $n = 450 \approx x[391]$. Therefore, a shift of $i = 400$ is selected to be used in the Monte Carlo simulations.

3.5.4. Monte Carlo Simulations

Again, Monte Carlo simulations will be used, this time to estimate the expected cosine similarity between the spectrum of the target after global whitening and the spectrum of the target after multi-area whitening.

For ZCA whitening, 18 different synthetic scans are created, each consisting of 200 pixels in total. The difference between the scans is the number of leather pixels, i.e., the ratio between the amount of Teflon pixels and leather pixels changes. The amount of leather pixels that are used in each scan are: 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100, 125, 150 and 175. Thus, the ratio between the amount of leather pixels and the total amount of pixels can be found in Table 3.2.

# Leather pixels	Leather to total pixels	Leather to Teflon pixels
2	$^{2}/_{200} = 0.010$	$^{2}/_{198} \approx 0.0101$
3	$^{3}/_{200} = 0.015$	$^{3}/_{197} \approx 0.0152$
4	$^{4}/_{200} = 0.020$	$^{4}/_{196} \approx 0.0204$
5	$^{5}/_{200} = 0.025$	$^{5}/_{195} \approx 0.0256$
6	$^{6}/_{200} = 0.030$	$^{6}/_{194} \approx 0.0309$
7	$^{7}/_{200} = 0.035$	$^{7}/_{193} \approx 0.0363$
8	$^{8}/_{200} = 0.040$	$^{8}/_{192} \approx 0.0417$
9	$^{9}/_{200} = 0.045$	$^{9}/_{191} \approx 0.0471$
10	$^{10}/_{200} = 0.050$	$^{10}/_{190} \approx 0.0526$
15	$^{15}/_{200} = 0.075$	$^{15}/_{185} \approx 0.0811$
20	$^{20}/_{200} = 0.100$	$^{20}/_{180} \approx 0.1111$
25	$^{25}/_{200} = 0.125$	$^{25}/_{175} \approx 0.1429$
50	$^{50}/_{200} = 0.250$	$^{50}/_{150} \approx 0.3333$
75	$^{75}/_{200} = 0.375$	$^{75}/_{125} = 0.6000$
100	$^{100}/_{200} = 0.500$	$^{100}/_{100} = 1.0000$
125	$^{125}/_{200} = 0.625$	$^{125}/_{75} \approx 1.6667$
150	$^{150}/_{200} = 0.750$	$^{150}/_{50} = 3.0000$
175	$^{175}/_{200} = 0.875$	$^{175}/_{25} = 7.0000$

Table 3.2: Number of leather pixels in a synthetic scan of 200 total pixels and the corresponding ratios to the total number of pixels in the scan and to the number of Teflon pixels in the scan.

Each scan gets whitened in its entirety using ZCA whitening, this is referred to as global ZCA whitening. Then, multi-area whitening is performed on the two parts of the scan. First, the pixels containing a Teflon spectrum get ZCA whitened, including the target pixel. Then, the pixels containing a leather spectrum get ZCA whitened. After this, the two resulting whitened matrices are combined to create the full whitened image. Now, two cosine similarities are calculated: the spectrum of the target pixel after global ZCA whitening and the spectrum of the target pixel after multi-area ZCA whitening are both compared to the negative Gaussian probability density function, as in Figure 3.2 or $-f(x)$ from Equation 3.1. As mentioned before, SAM is "relatively robust against variation in the total illumination intensity" [26], however, the intensity of a whitened spectrum usually lies between -3 and 3, and the intensity of the Gaussian probability density function almost reaches -2500, as it is not whitened. The cosine similarity is not able to handle this factor of 1000 and will give values around 0. Therefore, the Gaussian probability density function is multiplied by $^{1}/_{1000}$. This gives the following probability density function:

$$f(x) = \frac{A}{1000\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma^2}\right)^2} = \frac{150}{20\pi} e^{-\frac{1}{2}\frac{(x-300)^2}{100}}, \tag{3.2}$$

resulting in the same figure as Figure 3.2, but with the intensities on the y-axis divided by 1000. This is the spectrum used to calculate the cosine similarities of global ZCA whitening and multi-area ZCA whitening. 1000 Monte Carlo simulations are done to estimate the expected cosine similarity.

4

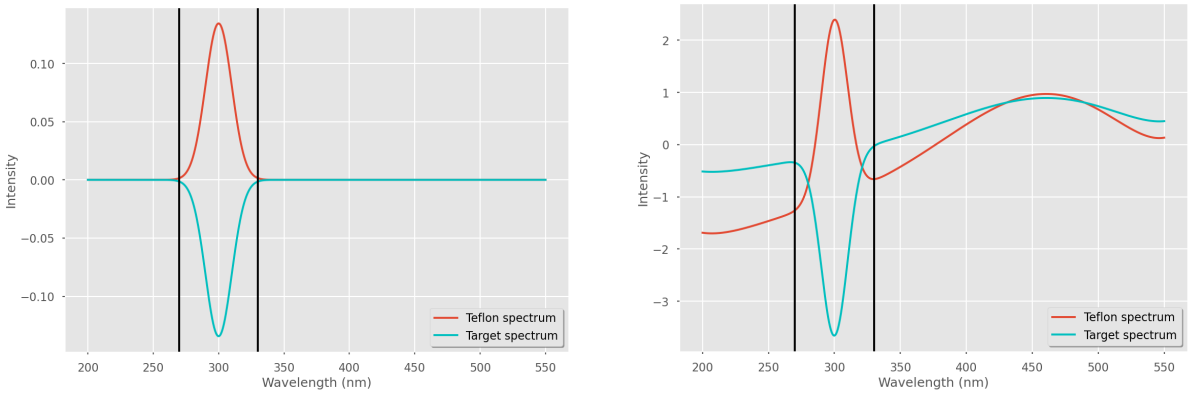
Results and Discussion

In this chapter, the results to the experiments described in Chapter 3 will be given and discussed. This includes average signal-to-noise ratio of a ZCA whitened target spectrum after 1000 Monte Carlo simulations, and the results of the multi-area whitening. The average cosine similarity between the target spectrum after global ZCA whitening and the Gauss from Equation 3.2, and the average cosine similarity between the target spectrum after multi-area ZCA whitening and the Gauss from Equation 3.2. But first, the spectra after ZCA whitening if shot noise was not added, are discussed.

4.1. Without Shot Noise

In order to clearly look at the effects of ZCA whitening, the step of adding noise, step 3 in Section 3.1, is left out. The process is then resumed at step 4. Not adding noise allows one to inspect the effects of whitening with ease.

First, a scan of only two pixels was created: one Teflon pixel and one target pixel. This results in a synthetic scan of size 2×548. This scan is whitened using ZCA whitening, both pixels and wavelengths were used as the observations. The result can be found in Figure 4.1.



(a) Two spectra of a 1×2 synthetic scan after ZCA whitening, the target spectrum and the Teflon spectrum, pixels used as observations.

(b) Two spectra of a 1×2 synthetic scan after ZCA whitening, the target spectrum and the Teflon spectrum, wavelengths used as observations.

Figure 4.1: Two spectra of a 1×2 synthetic scan without shot noise after ZCA whitening, the target spectrum and the Teflon spectrum, pixels and wavelengths used as observations.

In Figure 4.1a, the target spectrum clearly resembles the Gaussian from Figure 3.2. But most notable is that the whitened Teflon spectrum is the exact opposite of the target spectrum. This is due to the requirement that the whitened image should be row-wise centered. When using pixels as observations, this means that given any wavelength i, the sum of the intensities at wavelength i of all pixels in the scan should equal zero, i.e., for all i, $\mu_i \sum_{j=1}^{m} X_{i,j} = 0$, where m is the number pixels in the scan. Since there are only two pixels in the scan,

they should indeed be each other's opposite. The spectrum of a small amount of substance 'leaking' into the spectrum of the substrate is called the 'bleeding effect'. However, when using wavelengths as observations, this means that given any pixel i, the sum of the intensities at pixel i of all wavelengths in the scan should equal zero, i.e., for all i, $\mu_i \sum_{j=1}^{m} X_{i,j} = 0$, where m is the number wavelengths (spectral bands) in the scan. Thus the spectrum itself has an integral equal to 0, this can be seen in Figure 4.1b. Because of this, the spectrum outside of the Gaussian (the vertical lines in Figures 4.1a and 4.1b is not almost equal to 0, like in Figure 4.1a. Instead, an underlying shape can be made out, that seems to resemble the shape of the fitted Teflon polynomial from Figure 3.3. However, this claim is not proven, as it is outside of the scope of this research. But it is interesting to see what happens when all spectra are added together. When using pixels as observations, this equals zero at every wavelength, but when using wavelengths as observations, a much clearer resemblance to the Teflon polynomial from Figure 3.3 can be made out, see Figure 4.2.

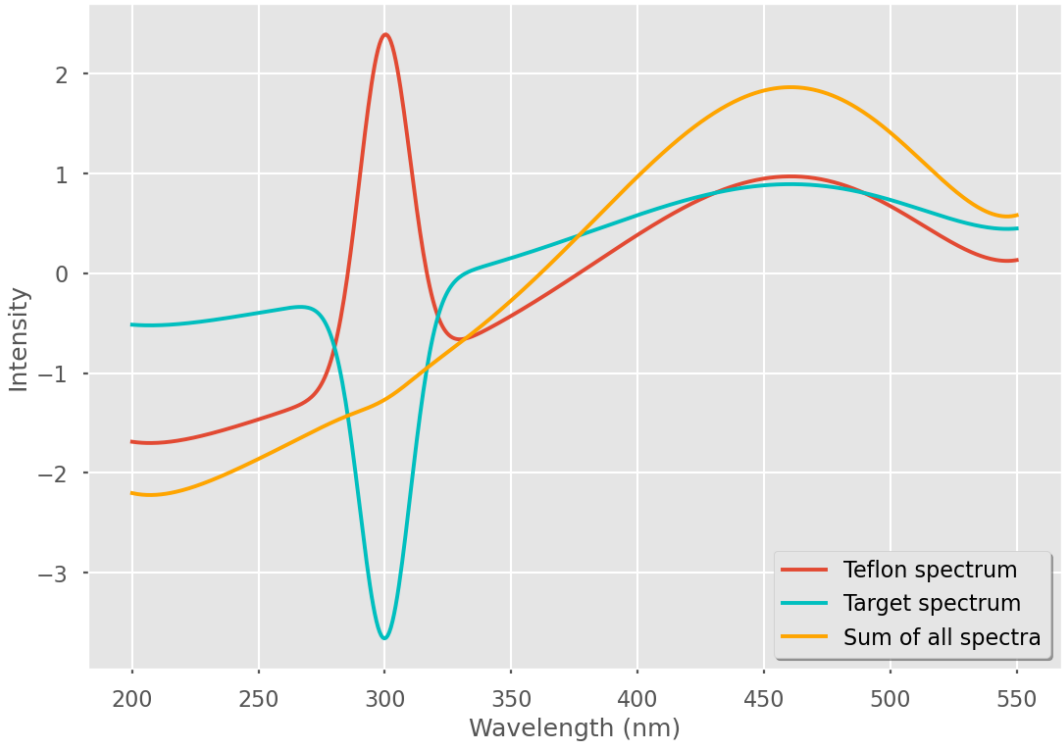
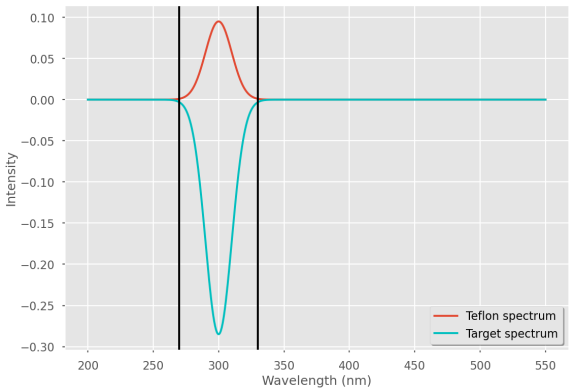


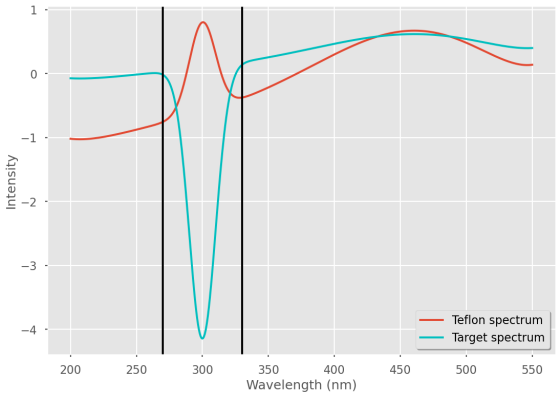
Figure 4.2: Two spectra of a 1×2 synthetic scan after ZCA whitening, the target spectrum and the Teflon spectrum and the sum of the two spectra, wavelengths used as observations.

In fact, if the Teflon spectrum is centered around zero, such that the integral of the spectrum equals zero, and it is multiplied by the maximum of the sum of all spectra divided by the maximum of the Teflon spectrum itself, i.e., the maximum of the Teflon spectrum is lined up with the maximum of the sum of all spectra, then the two are almost completely similar, with a cosine similarity of 0,99995. But a slight dip around 300 nm can be made out, so this procedure is also done on the target spectrum before whitening from Figure 3.3, resulting in a cosine similarity from 0,99982. As this is lower, but a dip can be seen around 300 nm, the suspicion follows that the sum of all spectra equals the average of the Teflon spectrum and the target spectrum before whitening, normalised as above. This results in a cosine similarity of 0,99999. Thus the sum of all spectra equals the average spectrum of all pixels in the scan before whitening, normalised. This explains the shape of the spectra of the whitened pixels.

Now, a scan of four pixels is created: three Teflon pixels and one target pixel. This results in a synthetic scan of size 4×548. This scan is whitened using ZCA whitening, both pixels and wavelengths were used as the observations. The result can be found in Figure 4.3.



(a) Two spectra of a synthetic with 4 pixels in total after ZCA whitening, the target spectrum and one of the three Teflon spectra, pixels used as observations.

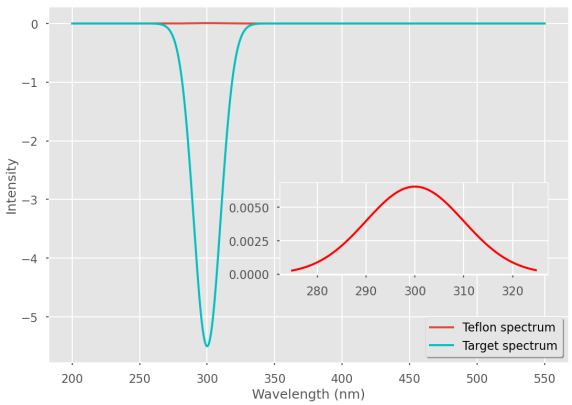


(b) Two spectra of a synthetic with 4 pixels in total after ZCA whitening, the target spectrum and one of the three Teflon spectra, wavelengths used as observations.

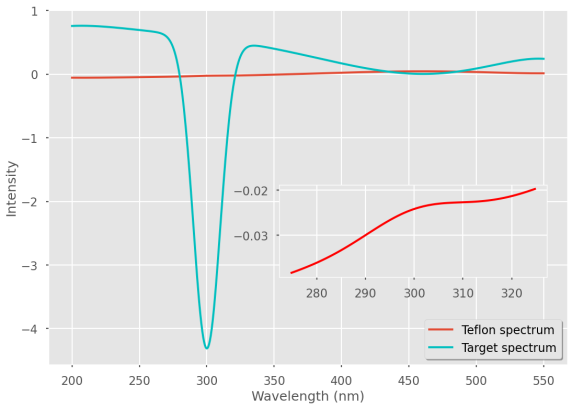
Figure 4.3: Two spectra of a synthetic with 4 pixels in total without shot noise after ZCA whitening, the target spectrum and one of the three Teflon spectra, pixels and wavelengths used as observations.

In Figure 4.3a, the target spectrum again clearly resembles the Gaussian from Figure 3.2, but with a higher intensity than in Figure 4.1a. But most notable is that the whitened Teflon spectrum is not the exact opposite of the target spectrum anymore. Although keeping a similar shape, the amplitude of the Gaussian is now much smaller, relative to the amplitude of the target spectrum. Since the sum of the intensities of all pixels at a given wavelength in the scan should equal zero, the burden to 'counteract' the target spectrum is now shared by the three Teflon pixels. The bleeding effect is smaller when more Teflon pixels are added to the scene. Note that since the input for all three Teflon pixels was exactly the same, the outputted spectra are also completely equal. And if these three spectra are added together, they indeed equal the opposite of the target spectrum, and everything adds op to zero. When using wavelengths as observations, similar consequences can be seen: the amplitude of the Gaussian in the target spectrum increases, whereas the amplitude of the Teflon spectrum decreases. Adding all spectra again gives a spectrum that very closely resembles the Teflon spectrum when normalised.

Lastly, a much larger scan is investigated, to see what happens when a scan is larger than 548 pixels. A scan of size 841×548 is used ($29 \times 29 \times 548$). The results can be found in Figure 4.4.



(a) Two spectra of a synthetic with 841 pixels in total after ZCA whitening, the target spectrum and one of the 840 Teflon spectra, pixels used as observations.



(b) Two spectra of a synthetic with 841 pixels in total after ZCA whitening, the target spectrum and one of the 840 Teflon spectra, wavelengths used as observations.

Figure 4.4: Two spectra of a synthetic with 841 pixels in total without shot noise after ZCA whitening, the target spectrum and one of the 840 Teflon spectra, pixels and wavelengths used as observations.

In Figure 4.4a, the target spectrum again clearly resembles a Gaussian, but with a higher intensity. Since there are now 840 Teflon pixels in the synthetic scan and one target pixel, the bleeding effect can barely be seen. Therefore, a blow-up is given of the whitened Teflon spectrum, such that the effect can be observed.

When using wavelengths as observations, the amplitude of the Gaussian of the whitened target spectrum has not increased much further than that of Figure 4.3b. Again, a blow-up is given of the whitened Teflon spectrum, such that the bleeding effect can be observed. Because of the high amount of pixels, the underlying shape of the Teflon spectrum is difficult to see, therefore, a figure without the target spectrum is given in Figure 4.5. This figure also contains the sum of all 841 spectra, divided by 841.

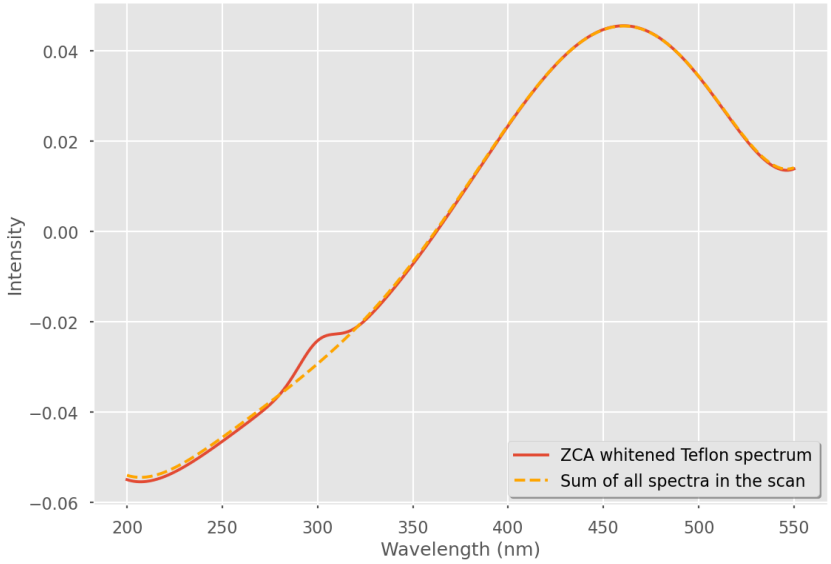
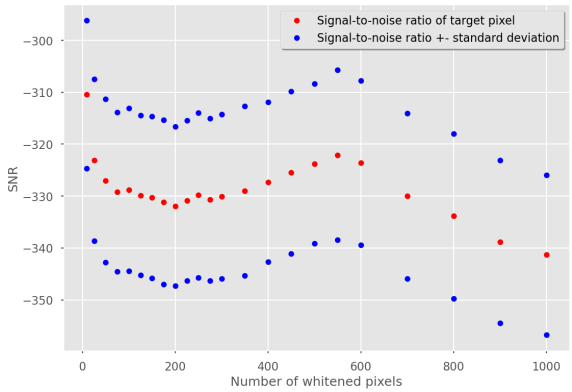


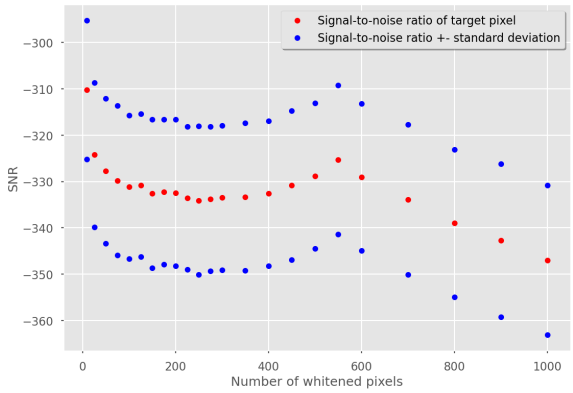
Figure 4.5: Two spectra of a synthetic scan with 841 pixels after ZCA whitening, the Teflon spectrum and the sum of all 841 spectra divided by 841, wavelengths used as observations.

4.2. Signal-to-Noise Ratio

Now, the step of adding noise, step 3 in Section 3.1, is not left out. If this noise would overpower the signal at 300 nm, the target substance cannot be identified. Thus, it is important that a signal can be made out when noise is present in the data. This relationship between the desired signal and the unwanted noise is expressed in the signal-to-noise ratio, which is calculated as described in Section 3.3. A higher signal-to-noise ratio means a higher ability to distinguish important patterns within the whitened spectra. Since there is reason to believe that there exists a connection between the amount of whitened pixels in the scene and the signal-to-noise ratio, the signal-to-noise ratio is calculated for different sizes of synthetic scans. It is expected that more pixels in the synthetic scan means a better signal-to-noise ratio, as the normally distributed noise cancels out. The average result of 1000 Monte Carlo simulations can be found in Figure 4.6.



(a) Signal-to-noise ratio of ZCA whitened target pixel, pixels used as observations.



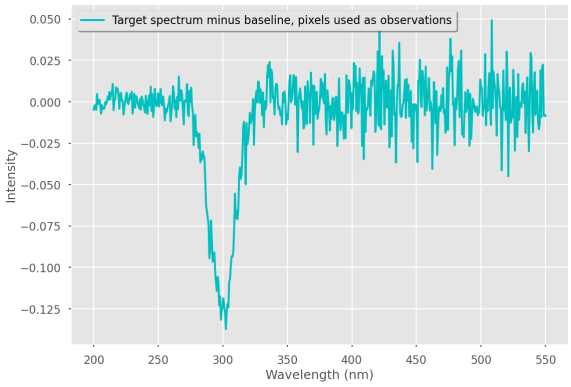
(b) Signal-to-noise ratio of ZCA whitened target pixel, wavelengths used as observations.

Figure 4.6: Signal-to-noise ratios for ZCA whitening against the number of pixels in the synthetic scan, with pixels and wavelengths used as observations, 1000 Monte Carlo simulations.

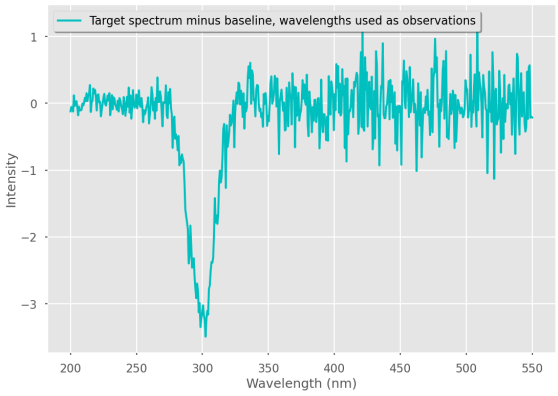
First of all, note that the signal-to-noise ratio's in both Figure 4.6a and Figure 4.6a are negative. This is because the signal that was calculated was the integral of the whitened target spectrum with the baseline subtracted, roughly between 270 nm and 300 nm. At these points, the spectrum lies below zero, thus giving a negative result. Normally, the signal-to-noise ratio is a positive number, where a higher signal-to-noise ratio thus means there is more signal than noise present. In our case, since the signal is negative, but the noise (σ_{RMS}) is positive, the signal-to-noise ratio is negative and thus a lower ratio indicates more signal to noise.

It is expected that the 1000 Monte Carlo simulations result in a set of values which are expected to be distributed as a normal distribution, but this needs to be proven. Therefore, a Kolmogorov-Smirnov test is done [6]. First, the data is normalised. The average ν_i of the 1000 signal-to-noise ratios is calculated: $\nu_i = \frac{1}{1000}\sum_{j=1}^{1000} SNR_{i,j}$, where $SNR_{i,j}$ is the signal-to-noise ratio of a scan of size i simulation j, $i \in \{9, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 350, 400, 450, 500, 550, 600, 700, 800, 900, 1000\}$. Then, the standard deviation σ_i of the 1000 signal-to-noise ratios of a scan of size i is calculated. Finally, the data is normalized according to the formula $\frac{SNR_{i,j}-\nu_i}{\sigma_i}$. If the data from scan size i was first distributed as a normal distribution with mean ν_i and standard deviation σ_i, then it is now distributed as a standard normal distribution, i.e., with mean 0 and standard deviation 1. Thus, the null hypothesis is that the data has a standard normal distribution. The p-values are calculated using the code in Appendix C.7, for pixels and wavelengths used as observations. The null hypothesis is rejected if the p-value is smaller than 0.05, which it not for any size i, nor pixels and wavelengths used as observations. Thus the hypothesis that the data is distributed as a normal distribution with mean ν_i and standard deviation σ_i is not rejected.

Secondly, note that the signal-to-noise ratio's when pixels are used as observations and when wavelengths are used as observations, look very similar. There are slight differences in how high the signal-to-noise ratio is, but the overall shape is the same. These differences are likely due to the subtraction of the baseline. The subtraction of the baseline ensures that the whitened target spectrum when using wavelengths as observations, looks highly similar to that of the whitened target spectrum when using pixels as observations. For example, look at the synthetic scan consisting of only two pixels again, this time with noise added. The target spectrum when wavelengths are used as observations contained an underlying shape, which was why the baseline was calculated and subtracted. This results in the two spectra in Figure 4.7.



(a) Target spectrum of a 1×2 synthetic scan after ZCA whitening with its baseline subtracted, pixels used as observations.



(b) Target spectrum of a 1×2 synthetic scan after ZCA whitening with its baseline subtracted, wavelengths used as observations.

Figure 4.7: Two target spectra of a 1×2 synthetic scan after ZCA whitening with their respective baselines subtracted, pixels and wavelengths used as observations.

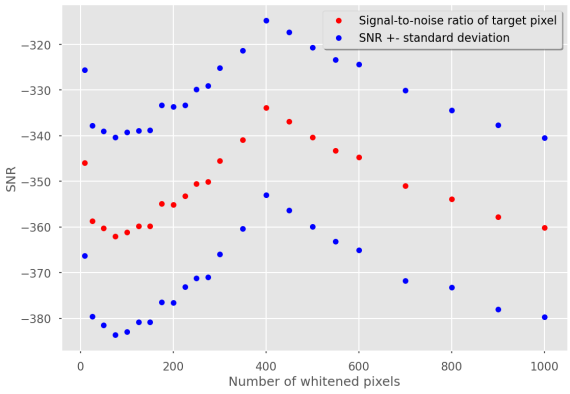
In Figure 4.7, it can be seen that the two whitened spectra look almost exactly equal, mostly differing in intensity. In fact, they have a cosine similarity of 0,99800. The slight difference is likely because the fitted baseline is not entirely perfect. This is also suspected to be the cause of the slight differences in the signal-to-noise ratio's of Figure 4.6. Using wavelengths as observations also gives a bigger amplitude, i.e., lower intensity, thus resulting in a slightly lower signal-to-noise ratio.

The last aspect of the signal-to-noise ratio's that needs to be mentioned, is the shape of the graph. At first, a decrease is observed until the scan contains 200 to 300 pixels. Then the signal-to-noise ratio increases until

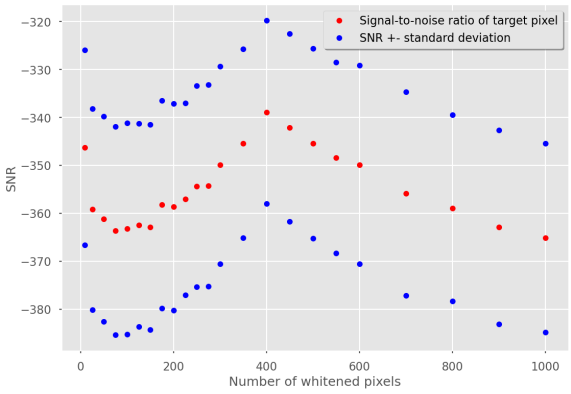
the scan contains 550 pixels. After 550 pixels, the signal-to-noise ratio seems to decrease linearly. The exact behaviour of the signal-to-noise ratio below 550 pixels is difficult to explain analytically, as this involves highly complicated matrix calculations, which is out of the scope of this thesis. What stands out besides the shape, is the point from where the signal-to-noise ratio's start to decrease seemingly linear, namely 550 pixels. It is mentioned in Section 3.2.1 that the empirical covariance matrix will always be of size 548×548 and can never be positive definite when there are less than 548 pixels in the scan, when pixels are used as observations. And when wavelengths are used as observations, the empirical covariance matrix will always be of size $n \times n$, where n is the number of pixels in the scan, and it can never be positive definite when there are more than 548 pixels in the scan. Therefore, it is hypothesised that the shape of the graph has to do with this number 548, the number of spectral bands or wavelengths that were used.

4.2.1. Cutting of the Spectrum

In the hope of proving the hypothesis that the shape of Figures 4.6a and 4.6b due to the number of wavelengths used, 548, the spectra are now cut off at 400 spectral bands. Meaning that the spectra used to create synthetic scans now range from 200 nm to roughly 455 nm. If the hypothesis is correct, then using 400 spectral bands instead of 548 should change the shape of Figures 4.6a and 4.6b such that the signal-to-noise ratio's now increase when getting closer to 400 whitened pixels per scan, and decrease when more than 400 pixels are used per scan. The result can be found in Figure 4.8 and the suspicion is confirmed.



(a) Signal-to-noise ratios for ZCA whitening against the number of pixels in the synthetic scan, if a spectrum contained only 400 wavelengths, pixels used as observations.

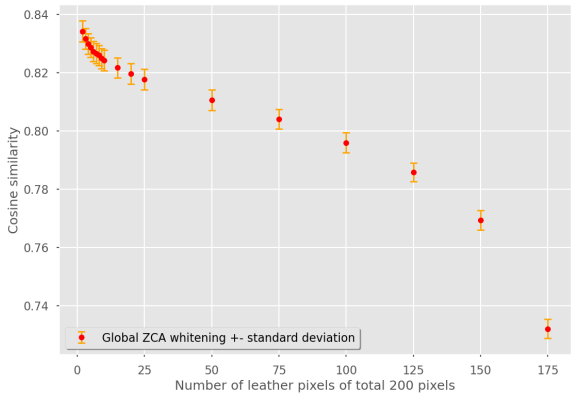


(b) Signal-to-noise ratios for ZCA whitening against the number of pixels in the synthetic scan, if a spectrum contained only 400 wavelengths, wavelengths used as observations.

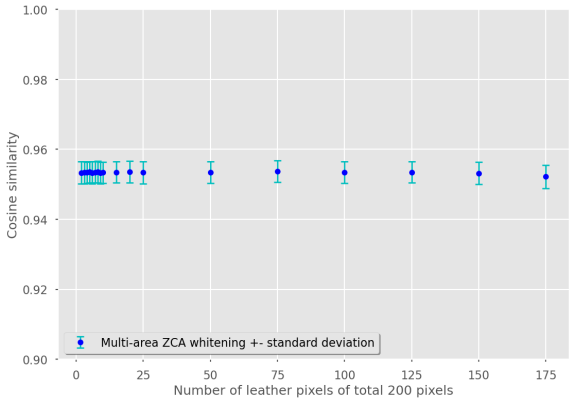
Figure 4.8: Signal-to-noise ratios for ZCA whitening against the number of pixels in the synthetic scan, if a spectrum contained only 400 wavelengths, with pixels and wavelengths used as observations, 1000 Monte Carlo simulations.

4.3. Multi-Area Whitening

When scanning an object that consists of multiple background materials, the spectra obtained after the whitening process will be affected by the intermixing of different input spectra prior to whitening, as can be seen in the Figure B.8. To avoid this intermixing, multi-area whitening was employed. In multi-area whitening, the entire scene is partitioned into distinct categories corresponding to the individual substrates. Each of these categories undergoes whitening independently. Next, all the whitened pixels are merged to form the final whitened image. This approach circumvents any potential impact of other materials on the spectra that might occur if all pixels were whitened simultaneously, which is important, as Stage Gate 11 B.V. uses the shape of the spectrum to identify substances. The resulting target spectra from both global ZCA whitening and multi-area ZCA whitening are compared to the negative Gaussian probability density function similar to the one in Figure 3.2, but multiplied by a factor $1/1000$, as in Equation 3.2. It is expected that the cosine similarity of the target spectrum from the multi-area whitened scan lies much closer to 1 than the cosine similarity of the target spectrum from the globally whitened scan, as the shape of the target spectrum is not affected by the presence of another substrate. The average result of 1000 Monte Carlo simulations can be found in Figures 4.9 and 4.10.



(a) Cosine similarity between the spectrum of the target pixel resulting from whitening the entire scan using global ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, pixels used as observations.

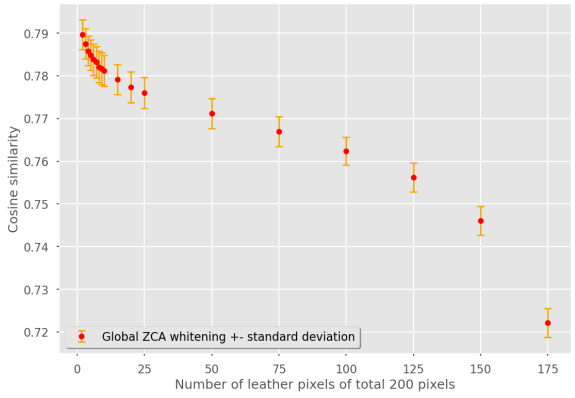


(b) Cosine similarity between the spectrum of the target pixel resulting from multi-area ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, pixels used as observations.

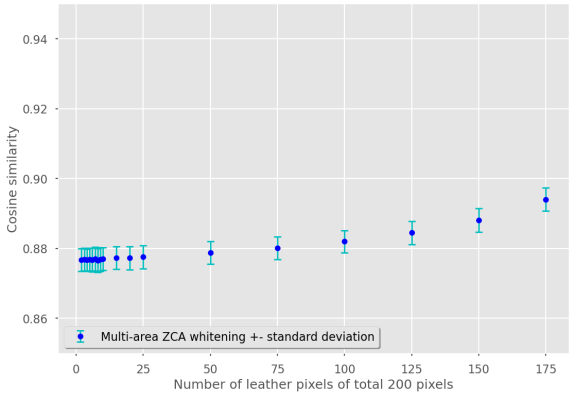
Figure 4.9: Cosine similarity between the spectrum of the target pixel resulting from whitening the entire scan using global ZCA whitening and the Gaussian probability density function from Equation 3.2, and cosine similarity between the spectrum of the target pixel resulting from multi-area ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, pixels used as observations.

It can be seen in Figure 4.9a, that the cosine similarity between the globally ZCA whitened target pixel and the Gaussian probability density function from Equation 3.2 decreases as the number of leather pixels in the scan increases. All spectra need to add up to 0 when pixels are used as observation, so when the number of leather pixels increases and the number of Teflon pixels decreases, the more the shape of the leather spectrum 'bleeds' into the spectrum of the target pixel, thus decreasing the cosine similarity to the Gaussian probability density function. But this 'bleeding' effect does not happen when all Teflon pixels are grouped together and whitened separately from the leather pixels, as can be seen in Figure 4.9b. Not only is the cosine similarity very stable, it is also much higher, when looking at the y-axis, due to the same reasons.

The same observations can be made when using wavelengths as observations, see Figure 4.10. The only notable differences are a slightly lower cosine similarity, and an increase in cosine similarity with the increasing number of leather pixels in the scan. Both of these differences are not necessarily because using pixels as observations is simply 'better', but more likely because the baseline was not subtracted and the underlying shape of the target spectrum has thus affected the cosine similarity.



(a) Cosine similarity between the spectrum of the target pixel resulting from whitening the entire scan using global ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, wavelengths used as observations.



(b) Cosine similarity between the spectrum of the target pixel resulting from multi-area ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, wavelengths used as observations.

Figure 4.10: Cosine similarity between the spectrum of the target pixel resulting from whitening the entire scan using global ZCA whitening and the Gaussian probability density function from Equation 3.2, and cosine similarity between the spectrum of the target pixel resulting from multi-area ZCA whitening and the Gaussian probability density function from Equation 3.2, 1000 Monte Carlo simulations, wavelengths used as observations.

Lastly, all results are given for two situations: the situation where the pixels are viewed as the observations and the wavelengths as variables, and the situation where the wavelengths are viewed as the observations and the pixels as variables. It is notable that, although not having the same shape after whitening, both methods have quite a similar results. The signal-to-noise ratio looks similar in Figure 4.6. This is because the baseline, i.e., the underlying shape of the spectrum outside of the Gaussian around 300 nm, was subtracted from the target spectrum when wavelengths are used as observations. But it must be noted that calculating this baseline might not be feasible when working with real data, i.e., not the synthetic data that was used in this thesis. When trying to classify an unknown target substance, it is not known beforehand where the peaks important to classification will lie in the spectrum. This makes it much harder to calculate and subtract a baseline. Since using wavelengths as observations also adds an underlying shape to the target spectrum depending on the substrate, target spectra from which no baseline is subtracted will look different from each other, depending on which substrate the target substance is placed upon. This makes it even more difficult to compare the whitened target spectra to a signature. Therefore, using wavelengths as observations makes less sense than using pixels as observations when using real data.

One final problem arises when using real data. The input image is of size $n \times m$, if n is the number of spectral bands or wavelengths, then the covariance matrix is of the size $n \times n$, i.e., it is always the same size 548×548. But when n is the number of pixels, the covariance matrix is of the size $n \times n$ and differs by the amount of pixels that are used. Since synthetic data is used, the number of pixels was not made that much larger than the number of wavelengths, and thus both methods work in a decent amount of time. But in reality, the standard scan of a shoe has 128 by 858 = 109824 pixels, meaning that the covariance matrix is of size 109824×109824 and contains about 12 billion entries. This is too large to work with and takes a lot of time, another reason why using wavelengths as observations is not practical.

5

Conclusions

From the results in Chapter 4, a few conclusions can be drawn about the ZCA whitening transformation.

First, what a target spectrum looks like before and after whitening. The synthetic data that was created in this thesis, used Teflon as a substrate. The target spectrum before whitening therefore looks similar to the spectrum of a piece of Teflon, but with a Gaussian probability density function subtracted from it around 300 nm. After ZCA whitening where pixels are used as observations, the spectrum of the target pixel is again a negative Gaussian probability density function. As the sum of all spectra should equal 0 at every spectral band i, this Gaussian probability density function 'bleeds' through to the whitened Teflon spectra, which have a peak around 300 nm, such that they all add up to equal the opposite intensity of the target pixel. A synthetic scan of size n consisted of $n-1$ Teflon pixels and one target pixel. Thus, the bigger the scan, the more Teflon pixels and therefore a smaller bleeding effect. However, when using wavelengths as observations, not the sum of all spectra should equal 0 at every spectral band i, but every spectrum itself should integrate to 0. This means that after ZCA whitening, there is still a negative peak visible around 300 nm, but there is another shape present in the target spectrum. The underlying shape is caused by the fact that all whitened spectra in the scan added together and divided by the number of pixels in the scan, equal the average of all unwhitened spectra from the original scan, but normalised such that they integrate to 0.

Secondly, when noise is added to the pixels in the synthetic scan, what is the relationship between the number of pixels in the scan and the signal-to-noise ratio. If the number of pixels in the scan is smaller than the number of spectral bands (548), then the behaviour of the signal-to-noise ratio is difficult to explain and considered out of the scope of the project. If the number of pixels in the scan is bigger than or equal the number of spectral bands, then the signal-to-noise ratio decreases as the number of pixels in the scan increases. This is a positive result, as the signal-to-noise ratio is negative in our case. As the number of pixels in the scan increases, the normally distributed noise cancels each other out, resulting in a better signal-to-noise ratio.

Then, another substrate was introduced to the scenario, resulting in the question of how a high contrast scenario influences the spectra. It could be clearly seen that the addition of a different spectrum to the scenario changed the shape of the whitened target spectrum. There has not been worked with the classification algorithm of Stage Gate 11 B.V., so the extent to which this negatively influences the classification is unknown, but potential negative effects can be negated by performing multi-area whitening. It was observed that the cosine similarity between the whitened target pixel and the negative Gaussian probability density function was much higher after multi-area ZCA whitening than after global ZCA whitening, meaning the shape of the spectrum more similar. As Stage Gate 11 B.V. uses the shape of the spectrum for the classification of a substance, this might result in more correct classifications, but it has to be noted that the amount of pixels used in the whitening transformation is smaller when using multi-area whitening.

In multi-area whitening, not all pixels in the scan are whitened at once, but different sections are whitened individually, resulting in smaller data matrices. This could mean a worse signal-to-noise ratio. But as for the question of cut-off criteria, no conclusion can be drawn for now. As mentioned above, there has not been worked with the classification algorithm of Stage Gate 11 B.V., and it is not possible to predict if the effect

of the changing shape of the target spectrum due to global whitening is greater than the effect of the better signal-to-noise ratio, or to predict if the effect of the stable shape of the target spectrum due to multi-area whitening is greater than the effect of the worse signal-to-noise ratio, when it comes to classifying a target pixel.

Finally, on whether to transpose the input image or not, i.e., whether to use pixels as observations or wavelengths as observation, it is concluded that using wavelengths as observations makes less sense than using pixels as observations when using real data, due to the difficulty in calculating the baseline when working with unknown substances, and the size that the covariance matrix can become, which becomes impractical to work with. Thus the data matrix should be of size $n \times m$, where n is the number of wavelengths, $n = 548$, and m is the number of pixels, which can vary per scan.

6

Recommendations

Due to the time constraint, it was not possible to use all whitening methods that were investigated. PCA, standardized PCA and Cholesky were deemed unsuitable for this project, since the position of the negative Gausian probability density function is important for classification, but this was not the problem with CCA and standardized ZCA. These two methods, although very interesting, could not be effectively implemented within the allotted time frame. Further investigating of standardized ZCA and CCA could give interesting results, but CCA requires sets of signatures and for the method to be tested on multiple types of target substances. From Table 2.1 it can be seen that CCA is actually very similar to standardized ZCA, as $W^X_{CCA} = Q^T_X W_{ZCA-std}$.

Although it was concluded that using pixels as observations was easier, especially when working with real data, when working with wavelengths as observations, one more problem arises. Although the empirical covariance matrix is an unbiased estimator, when the wavelengths are used as observations and the number of pixels is much higher than the number of wavelengths, $m \gg n$, the empirical covariance matrix is not a good estimator and is estimated with a lot of error [16]. A possible solution to this is to "shrink" the empirical covariance matrix to a more structured estimator T. The estimator T will have less variance, but introduce a bias. This is called the bias-variance trade-off. The new estimator $\tilde{\Sigma}$, will be a convex combination of the two estimators: $\tilde{\Sigma} = \delta T + (1-\delta)\hat{\Sigma}$, where $\delta \in [0,1]$ is the shrinkage intensity. δ^* is the optimal shrinkage estimator for which the squared error loss risk function is minimal [22]. If wavelengths are used as observations and scans much larger than 1000 pixels are used, it is recommended to shrink the covariance matrix. A lot of research has been done on shrinkage of the covariance matrix by Ledoit and Wolf [16–18].

When considering the study of the multi-area whitening within the current scope, it can be observed that the relatonship between parameters is not fully explored at this stage. In this study, the total number of pixels in the scene is kept constant, and thus the number of Teflon pixels and leather pixels vary with each experiment. Consequently, the effect of the number of pixels on the signal-to-noise ratio is not independently observed. It is thus currently not possible to conclude whether a small area, which is locally whitened, will show a sufficient signal-to-noise ratio of the target spectrum to allow for a good cosine similarity. It is recommended for completeness to conduct a further thorough investigation of potential interdependencies. The methods described in Chapter 3 may be used to design such a further experiment.

Bibliography

[1] Bolla, M. (2021). *Multivariate Statistics: Properties of the Multivariate Normal Distribution*. Budapest University of Technology and Economics, Institute of Mathematics. https://math.bme.hu/~marib/tobbvalt/.

[2] Bradaschia, F. (2013). Components of Electromagnetic Spectrum. https://www.radio2space.com/components-of-electromagnetic-spectrum/.

[3] Busch, K. and Busch, M. (2018). Chapter 3 - Light Polarization and Signal Processing in Chiroptical Instrumentation. In Polavarapu, P., editor, *Chiral Analysis (Second Edition)*, pages 73–151. Elsevier. https://doi.org/10.1016/B978-0-444-64027-7.00003-3.

[4] de Groot, R. (2022). Een Gerichte Studie naar Contaminatie bij de Fabricage van Bomschoenen. *Unpublished confidential document*.

[5] Edmund Optics (2020). Hyperspectral and Multispectral Imaging. https://www.edmundoptics.eu/knowledge-center/application-notes/imaging/hyperspectral-and-multispectral-imaging/.

[6] Encyclopedia of Mathematics (2020). Kolmogorov-Smirnov test. http://encyclopediaofmath.org/index.php?title=Kolmogorov%E2%80%93Smirnov_test&oldid=22660.

[7] GIS Geography (2023). Multispectral vs Hyperspectral Imagery Explained. https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained/.

[8] Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering*. Springer, New York, NY. https://doi.org/10.1007/978-0-387-21617-1.

[9] Hummel, R. and Dubroca, T. (2006). Differential Reflectance Spectroscopy in Analysis of Surfaces. *Encyclopedia of Analytical Chemistry*. https://doi.org/10.1002/9780470027318.a2504.

[10] Instruments S.A. Inc. JOBIN YVON/SPEX Division (1994). *Guide for Spectroscopy*. Horiba Scientific, 3880 Park Avenue, Edison, NJ 08820-3097, USA.

[11] Jendoubi, T. and Strimmer, K. (2019). A Whitening Approach to Probabilistic Canonical Correlation Analysis for Omics Data Integration. *BMC Bioinformatics*, 20(15). https://doi.org/10.1186/s12859-018-2572-9.

[12] Karabiber, F. (2021). Cosine Similarity. https://www.learndatasci.com/glossary/cosine-similarity/.

[13] Kessy, A., Lewin, A., and Strimmer, K. (2018). Optimal Whitening and Decorrelation. *The American Statistician*, 72(4):309–314. https://doi.org/10.1080/00031305.2016.1277159.

[14] Lay, S. (2014). *Analysis with an Introduction to Proof*. Pearson Education, 2014 custom edition.

[15] Leboran, V., Garcia-Diaz, A., Fdez-Vidal, X., and Pardo, X. (2017). Dynamic Whitening Saliency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5). https://doi.org/10.1109/TPAMI.2016.2567391.

[16] Ledoit, O. and Wolf, M. (2004). "Honey, I Shrunk the Ssample Covariance Matrix". *Journal of Portofolio Management*, 30(4):110–119. https://doi.org/10.3905/jpm.2004.110.

[17] Ledoit, O. and Wolf, M. (2015). Spectrum estimation: A unified framework for covariance matrix estimation and PCA in large dimensions. *Journal of Multivariate Analysis*, 139:360–384. http://dx.doi.org/10.1016/j.jmva.2015.04.006.

[18] Ledoit, O. and Wolf, M. (2019). The power of (non-)linear shrinking: A review and guide to covariance matrix estimation. Working Paper 323, University of Zurich, Department of Economics. https://doi.org/10.5167/uzh-170642.

[19] Nandram, A. (2023). Amper wachtrijen meer op schiphol, maar is de luchthaven klaar voor een probleemloze zomer? https://www.volkskrant.nl/nieuws-achtergrond/amper-wachtrijen-meer-op-schiphol-maar-is-de-luchthaven-klaar-voor-een-probleemloze-zomer~b0afe9a0/?

[20] Pourahmadi, M. (2011). Covariance Estimation: The GLM and Regularization Perspectives. *Statistical Science*, 26(3). https://www.jstor.org/stable/23059137.

[21] Satink, R. (2023). Personal communication.

[22] Schäfer, J. and Strimmer, K. (2005). A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1). https://doi.org/10.2202/1544-6115.1175.

[23] Schottky, W. (1918). Über spontane Stromschwankungen in verschiedenen Elektrizitätsleitern. *Annalen der Physik*, 362(23). https://doi.org/10.1002/andp.19183622304.

[24] Severn, K. (2023). *Multivariate Statistics*. University of Nottingham. https://rich-d-wilkinson.github.io/MATH3030/index.html.

[25] Sisson, M. (2014). Richard Reid. https://www.britannica.com/biography/Richard-Reid.

[26] Weyermann, J., Schläpfer, D., Hueni, A., Kneubühler, M., and Schaepman, M. (2009). Spectral Angle Mapper (sam) for anisotropy class indexing in imaging spectrometry data. *Imaging Spectrometry*, XIV(74570B). https://doi-org.tudelft.idm.oclc.org/10.1117/12.825991.

[27] Yong, H. and Zhang, L. (2022). An Embedded Feature Whitening Approach to Deep Neural Network Optimization. European Conference on Computer Vision. https://www.ecva.net/papers/eccv_2022/papers_ECCV/html/358_ECCV_2022_paper.php.

A

The Empirical Covariance Matrix: Additional Proofs

In this first appendix, additional proofs will be provided to the ones given in Section 2.3, where some proofs used the true covariance matrix Σ, instead of the empirical covariance matrix $\hat{\Sigma}$, namely Theorem 2.3.2, 2.3.3, 2.3.4. First note that Definition 2.3.1 needs to be updated using the empirical covariance matrix.

Definition A.0.1. Let $X \in \mathbb{R}^{n\times m}$ be a row-wise centered data matrix (i.e., the average of every row equals 0) with empirical covariance matrix $\hat{\Sigma}$. A whitening matrix of X is a matrix $\hat{W}$ such that $\hat{W}^T\hat{W} = \hat{\Sigma}^{-1}$.

If $\hat{W}$ and $\hat{\Sigma}$ are invertible, which they are assume to be, this is equivalent to $\hat{W}\hat{\Sigma}\hat{W}^T = I$, because

$$\begin{aligned} \hat{W}^T\hat{W} &= \hat{\Sigma}^{-1} \\ \hat{\Sigma}\hat{W}^T\hat{W} &= I \\ \hat{\Sigma}\hat{W}^T &= \hat{W}^{-1} \\ \hat{W}\hat{\Sigma}\hat{W}^T &= I. \end{aligned}$$

Now, it is proven that Theorem 2.3.2 also holds when using the empirical covariance matrix $\hat{\Sigma}$.

Theorem A.0.2. *Let $X \in \mathbb{R}^{n\times m}$ be a row-wise centered data matrix with empirical covariance matrix $\hat{\Sigma}_X$. Let $\hat{W}$ be a whitening matrix of X. Then, the whitened data matrix $Y = \hat{W}X$ also has the average of every row equal to 0, and empirical covariance matrix I.*

Proof. For the row average, note that the row average of row i of X equals 0, i.e., $\sum_{j=1}^{m} X_{i,j} = 0$, for all $i = 1,\dots,n$. The aim is to prove that $\sum_{j=1}^{m}\left(\hat{W}X\right)_{i,j} = 0$ for all $i = 1,\dots,n$. Start by observing what an element of $\hat{W}X$ at row i and column j looks like:

$$\begin{aligned} \left(\hat{W}X\right)_{i,j} &= \sum_{k=1}^{n} \hat{W}_{i,k}X_{k,j}, \\ \sum_{j=1}^{m}\left(\hat{W}X\right)_{i,j} &= \sum_{j=1}^{m}\sum_{k=1}^{n} \hat{W}_{i,k}X_{k,j} = \sum_{k=1}^{n}\sum_{j=1}^{m} \hat{W}_{i,k}X_{k,j} \\ &= \sum_{k=1}^{n}\left(\hat{W}_{i,k}\left(\sum_{j=1}^{m} X_{k,j}\right)\right) = 0. \end{aligned}$$

For the covariance, the following equalities are found:

$$\begin{aligned} \hat{\Sigma}_Y &= \frac{1}{m-1}YY^T = \frac{1}{m-1}\hat{W}X\left(\hat{W}X\right)^T \\ &= \frac{1}{m-1}\hat{W}XX^T\hat{W}^T = \hat{W}\hat{\Sigma}_X\hat{W}^T = I. \end{aligned}$$

□

Since the covariance matrix is a real and symmetric matrix, an eigendecomposition was performed, and $\Sigma = VDV^T$ resulted. Using the data, the eigendecomposition will be performed on $\hat{\Sigma}$, and thus $\hat{\Sigma} = \hat{V}\hat{D}\hat{V}^T$, where $\hat{V}$ is a orthogonal matrix with the orthonormal eigenvectors of $\hat{\Sigma}$ as its columns, and $\hat{D}$ a diagonal matrix with the corresponding eigenvalues as its diagonal values. The eigenvalues are put in descending order, such that $\hat{D}_{i,i} \geq \hat{D}_{i+1,i+1}$.

Theorem A.0.3. *If $\hat{W} = U_1\hat{\Sigma}^{-1/2} = U_1\hat{V}\hat{D}^{-1/2}\hat{V}^T$ with U_1 any orthogonal matrix of size $n \times n$, then $\hat{W}$ is a whitening matrix.*

Proof. It is easy to see that the proof of Theorem 2.3.3 still holds when W, Σ, V and D are replaced by $\hat{W}$, $\hat{\Sigma}$, $\hat{V}$ and $\hat{D}$, respectively. □

Another possibility was to decompose the covariance matrix Σ into the correlation matrix P and diagonal covariance matrix S, such that $\Sigma = S^{1/2}PS^{1/2}$. Now, the empirical covariance matrix $\hat{\Sigma}$ is decomposed, such that $\hat{\Sigma} = \hat{S}^{1/2}\hat{P}\hat{S}^{1/2}$. We interpret the input matrix $X \in \mathbb{R}^{n\times m}$ as a matrix of row vectors with mean zero, such that $X = (X_1, X_2, \ldots, X_n)^T$ with $X_1, X_2, \ldots, X_n \in \mathbb{R}^{1\times m}$ and $\mathbb{E}[X_i] = 0$ for $i = 1, \ldots, n$. We find the following diagonal empirical covariance matrix $\hat{S}$ and empirical correlation matrix $\hat{P}$:

$$\hat{S} = \begin{bmatrix} \hat{\sigma}^2_{X_1} & 0 & \ldots & 0 \\ 0 & \hat{\sigma}^2_{X_1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \hat{\sigma}^2_{X_n} \end{bmatrix} \text{ and } \hat{P} = \frac{1}{m-1}\begin{bmatrix} \frac{X_1X_1^T}{\hat{\sigma}_{X_1}\hat{\sigma}_{X_1}} & \frac{X_1X_2^T}{\hat{\sigma}_{X_1}\hat{\sigma}_{X_2}} & \cdots & \frac{X_1X_n^T}{\hat{\sigma}_{X_1}\hat{\sigma}_{X_n}} \\ \frac{X_1X_2^T}{\hat{\sigma}_{X_1}\hat{\sigma}_{X_2}} & \frac{X_2X_2^T}{\hat{\sigma}_{X_2}\hat{\sigma}_{X_2}} & \cdots & \frac{X_2X_n^T}{\hat{\sigma}_{X_2}\hat{\sigma}_{X_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{X_1X_n^T}{\hat{\sigma}_{X_1}\hat{\sigma}_{X_n}} & \frac{X_2X_n^T}{\hat{\sigma}_{X_2}\hat{\sigma}_{X_n}} & \cdots & \frac{X_nX_n^T}{\hat{\sigma}_{X_n}\hat{\sigma}_{X_n}} \end{bmatrix}.$$

Note that the diagonal elements of $\hat{P}$, $\frac{X_iX_i^T}{\hat{\sigma}_{X_i}\hat{\sigma}_{X_i}}$ should equal 1. Therefore, $\hat{\sigma}_{X_i} = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(X_{i,j})^2}$ was used, where $X_{i,j} = Im_{i,j} - \mu_i$. The sum is divided by $m-1$ instead of m. If one would divided by m, then $\hat{P}_{i,i}$ would not equal 1. Using this, the same decomposition is indeed found:

$$\hat{\Sigma} = \hat{S}^{1/2}\hat{P}\hat{S}^{1/2} = \begin{bmatrix} X_1X_1^T & X_1X_2^T & \ldots & X_1X_n^T \\ X_1X_2^T & X_2X_2^T & \ldots & X_2X_n^T \\ \vdots & \vdots & \ddots & \vdots \\ X_1X_n^T & X_2X_n^T & \ldots & X_nX_n^T \end{bmatrix}.$$

Theorem A.0.4. *If $\hat{W} = U_2\hat{P}^{-1/2}\hat{S}^{-1/2}$ with U_2 any orthogonal matrix of size $n \times n$, then $\hat{W}$ is a whitening matrix.*

Proof. It is easy to see that the proof of Theorem 2.3.4 still holds when W, Σ, S and P are replaced by $\hat{W}$, $\hat{\Sigma}$, $\hat{S}$ and $\hat{P}$, respectively. □

We found that the cross-covariance Φ and cross-correlation Ψ matrices between the whitened matrix $Y = WX$ and the original data centered matrix X were linked to the rotation matrices U_1 and U_2. This also holds for the empirical cross-covariance matrix $\hat{\Phi}$ and the empirical correlation matrix $\hat{\Psi}$:

$$\begin{aligned} \hat{\Phi} &= \frac{1}{m-1}\hat{W}XX^T = \hat{W}\hat{\Sigma} = \hat{U}_1\hat{\Sigma}^{-1/2}\hat{\Sigma} = \hat{U}_1\hat{\Sigma}^{1/2}, \\ \hat{\Psi} &= \hat{\Phi}\hat{S}^{-1/2} = U_1\hat{\Sigma}^{1/2}\hat{S}^{-1/2} = U_2\hat{R}\hat{\Sigma}^{1/2}\hat{S}^{-1/2} = U_2\hat{P}^{-1/2}\hat{S}^{-1/2}\hat{\Sigma}\hat{S}^{-1/2} = U_2\hat{P}^{-1/2}\hat{P} = U_2\hat{P}^{1/2}, \end{aligned}$$

where $\hat{R} = \hat{P}^{-1/2}\hat{S}^{-1/2}\hat{\Sigma}^{1/2}$ and $U_1 = U_2\hat{R}$.

For PCA whitening, choose U_1 from Theorem 2.3.3 to be $\hat{V}^T$, i.e., $\hat{W} = \hat{V}^T\hat{\Sigma}^{-1/2} = \hat{V}^T\hat{V}\hat{D}^{-1/2}\hat{V}^T = \hat{D}^{-1/2}\hat{V}^T$. The matrix $\hat{V}^T$ rotates the data, such that the points are projected upon the principal components. This means that the data is now decorrelated, but not yet with variance 1. This is because

$$\left(\hat{V}^TX\right)\left(\hat{V}^TX\right)^T = \hat{V}^TXX^T\hat{V} = (m-1)\hat{V}^T\hat{\Sigma}\hat{V} = (m-1)\hat{V}^T\hat{V}\hat{D}\hat{V}^T\hat{V} = (m-1)\hat{D}.$$

To get the variances to 1, multiply by $\hat{D}^{-1/2}$ and find

$$\hat{\Sigma}_Y = \frac{1}{m-1}\left(\hat{D}^{-1/2}\hat{V}^TX\right)\left(\hat{D}^{-1/2}\hat{V}^TX\right)^T = \frac{1}{m-1}\hat{D}^{-1/2}\left(\hat{V}^TX\right)\left(\hat{V}^TX\right)^T\hat{D}^{-1/2} = (m-1)\frac{1}{m-1}\hat{D}^{-1/2}\hat{D}\hat{D}^{-1/2} = I.$$

It is easy to see that in Theorems 2.3.6, 2.3.7, 2.3.10 and 2.3.11 and their proofs, all random matrices can be replaced by their estimated counterparts, and the proofs will still hold.

B

Figures

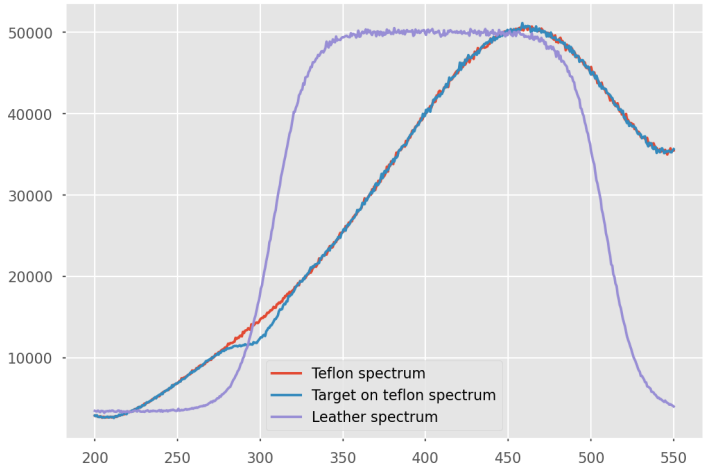
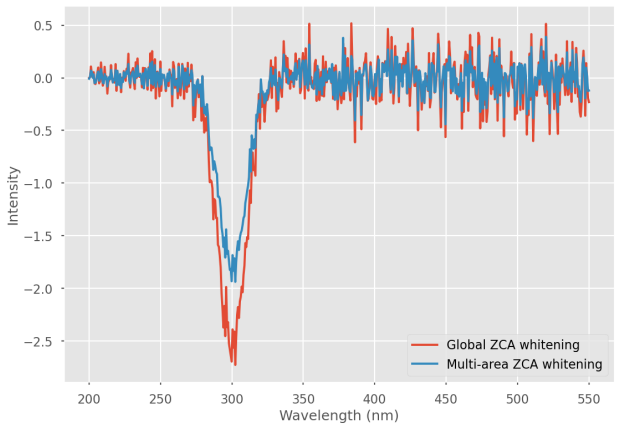
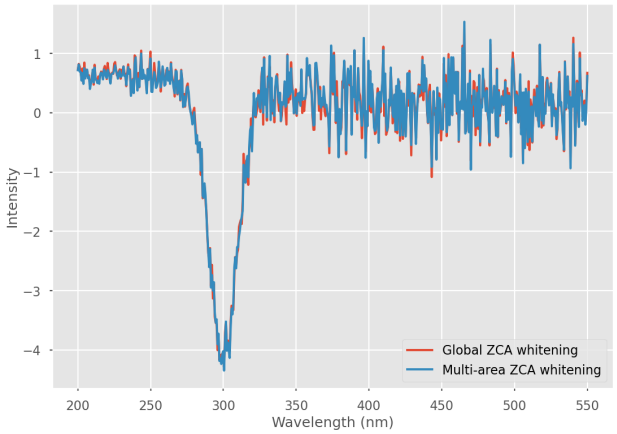


Figure B.1: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[325] \approx 407,95$ to create the leather spectrum.



(a) Pixels used as observations.



(b) Wavelengths used as observations.

Figure B.2: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[325] \approx 407,95$ to create the leather spectrum, pixels and wavelengths used as observations.

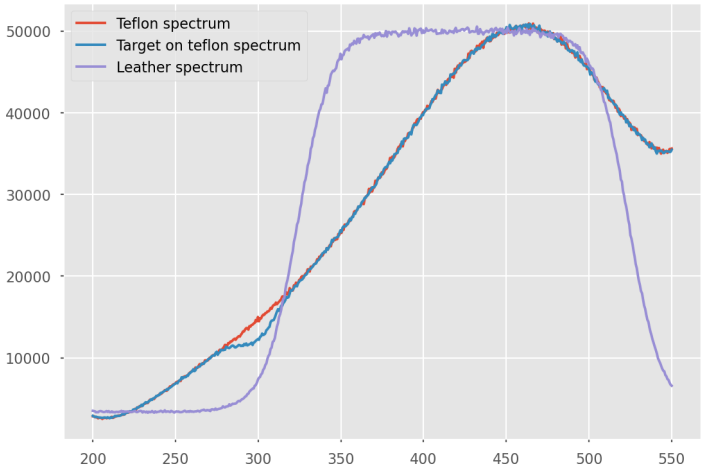
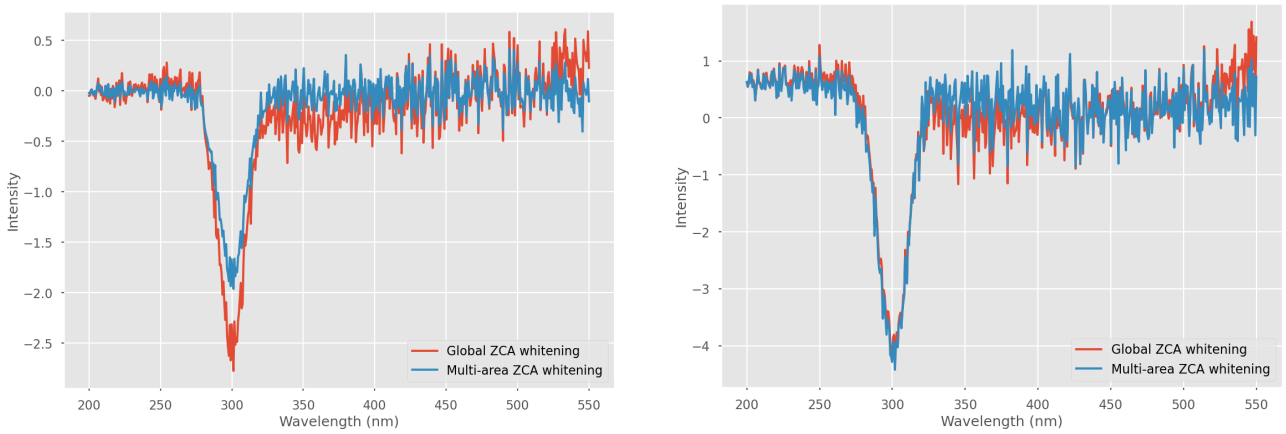


Figure B.3: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[350] \approx 423,95$ to create the leather spectrum.



(a) Pixels used as observations.

(b) Wavelengths used as observations.

Figure B.4: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[350] \approx 423,95$ to create the leather spectrum, pixels and wavelengths used as observations.

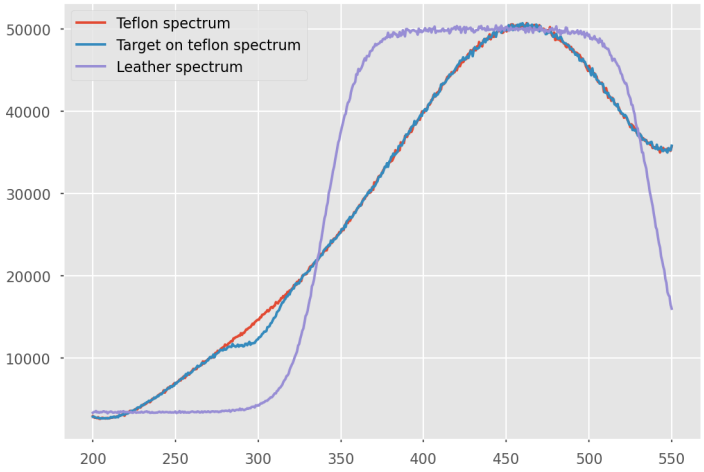
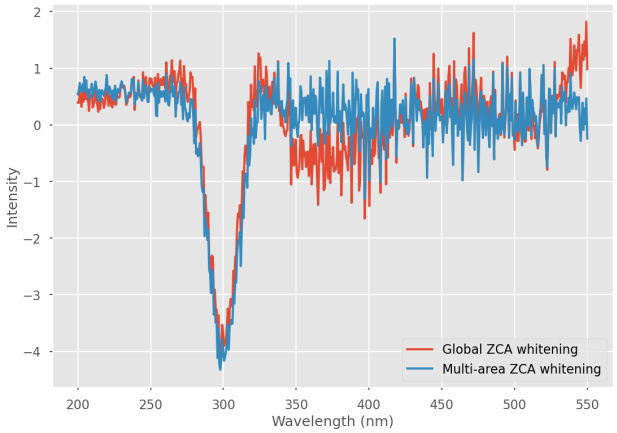


Figure B.5: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[375] \approx 439,95$ to create the leather spectrum.

(a) Pixels used as observations.



(b) Wavelengths used as observations.

Figure B.6: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[375] \approx 439,95$ to create the leather spectrum, pixels and wavelengths used as observations.

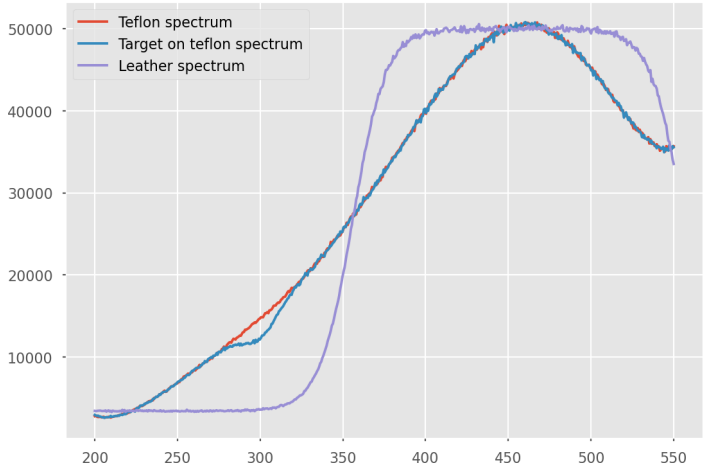
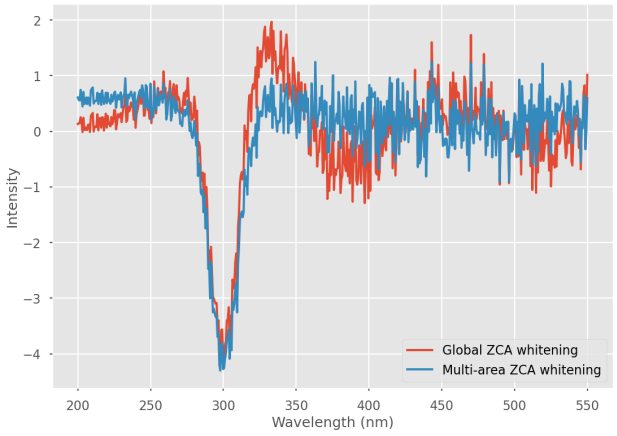


Figure B.7: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[400] \approx 455,94$ to create the leather spectrum.

(a) Pixels used as observations.



(b) Wavelengths used as observations.

Figure B.8: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[400] \approx 455,94$ to create the leather spectrum, pixels and wavelengths used as observations.

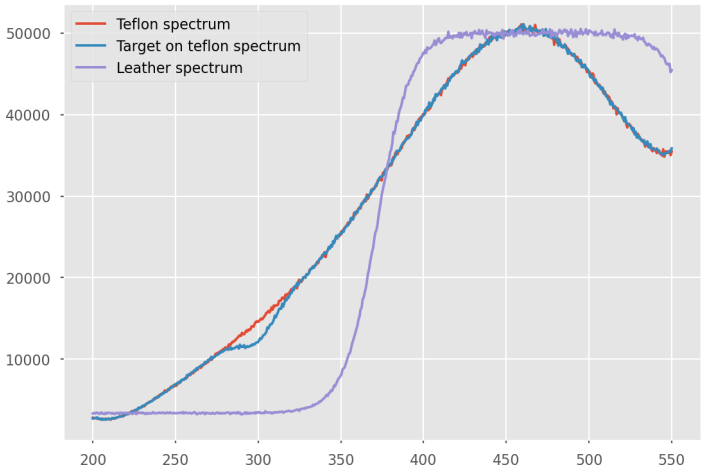
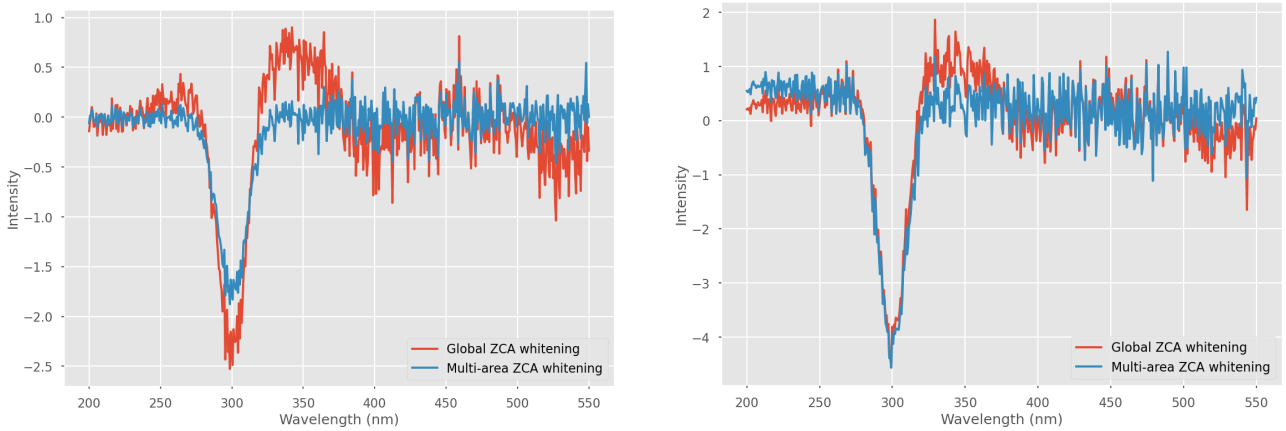


Figure B.9: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[425] \approx 471,94$ to create the leather spectrum.



(a) Pixels used as observations.

(b) Wavelengths used as observations.

Figure B.10: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[425] \approx 471,94$ to create the leather spectrum, pixels and wavelengths used as observations.

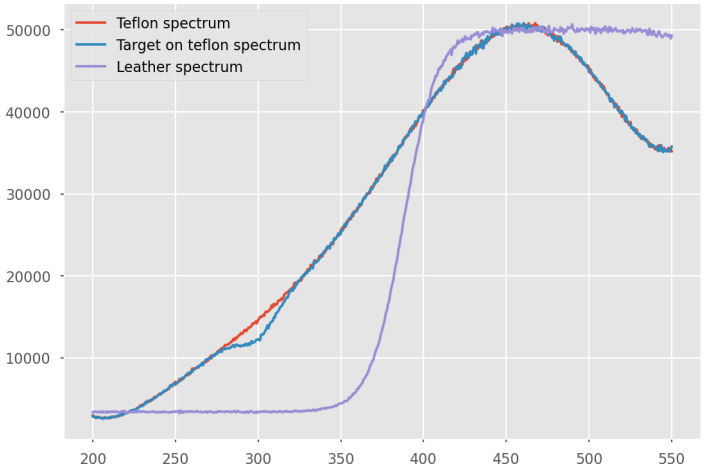
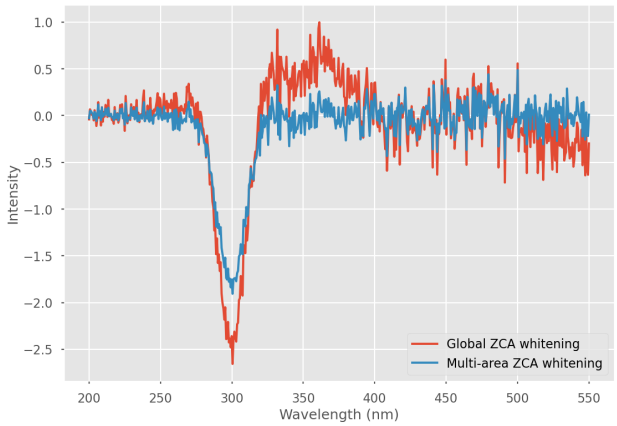
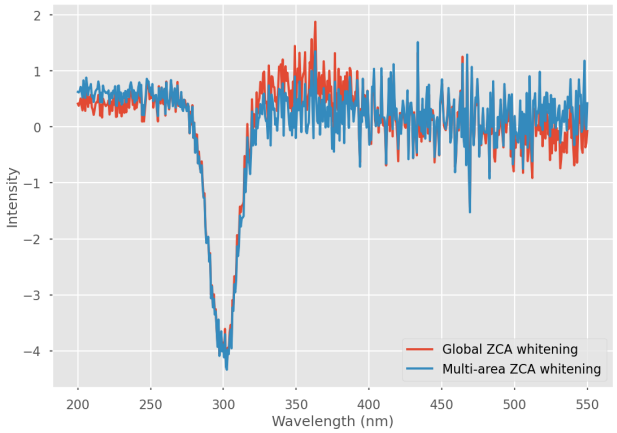


Figure B.11: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[450] \approx 487,93$ to create the leather spectrum.



(a) Pixels used as observations.



(b) Wavelengths used as observations.

Figure B.12: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[450] \approx 487{,}93$ to create the leather spectrum, pixels and wavelengths used as observations.

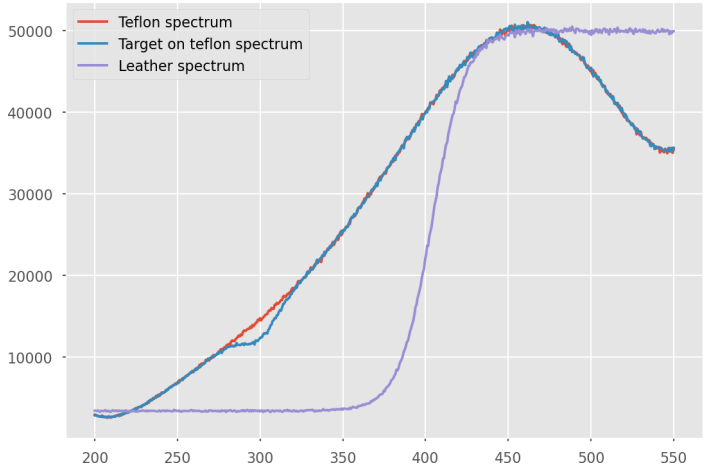
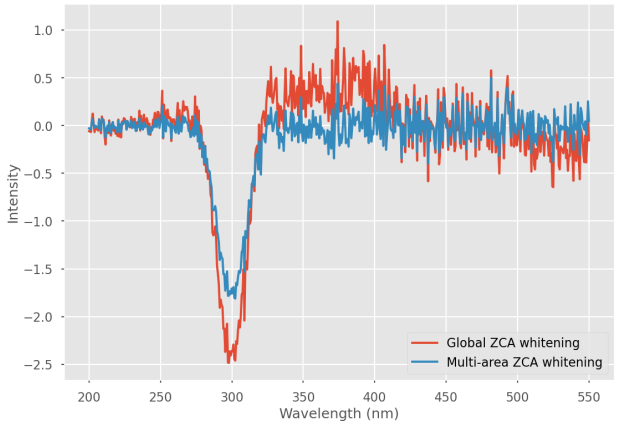
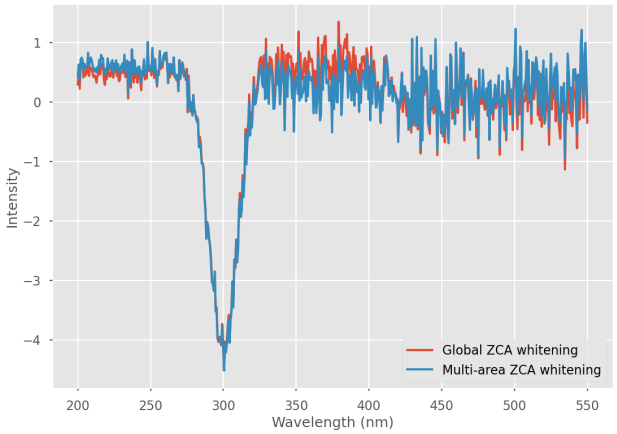


Figure B.13: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[475] \approx 503{,}93$ to create the leather spectrum.



(a) Pixels used as observations.



(b) Wavelengths used as observations.

Figure B.14: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[475] \approx 503{,}93$ to create the leather spectrum, pixels and wavelengths used as observations.

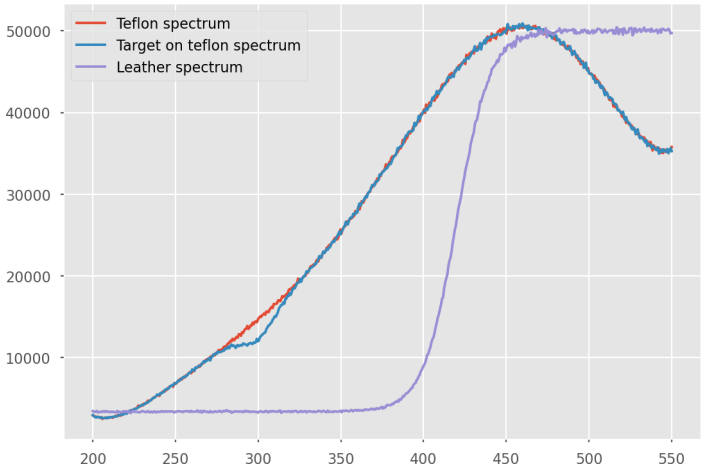
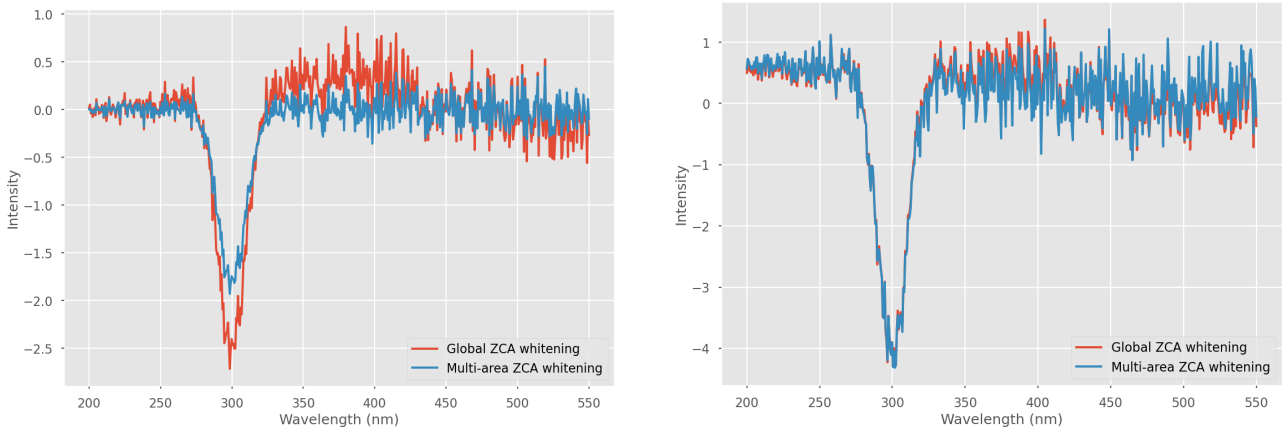


Figure B.15: Spectra of unwhitened pixels from the scenario in Figure 3.10, using $n = x[500] \approx 519,93$ to create the leather spectrum.



(a) Pixels used as observations.

(b) Wavelengths used as observations.

Figure B.16: Spectra of whitened pixels from the scenario in Figure 3.10, using $n = x[500] \approx 519,93$ to create the leather spectrum, pixels and wavelengths used as observations.

C

Code

C.1. whitening_math_functions

```
import numpy as np

# Code to center the scan
def center(scan):
    if scan.ndim == 1:
        meanPoint = scan.mean()
    else:
        meanPoint = scan.mean(axis = 1)
        meanPoint = meanPoint[:, np.newaxis]
    scan_centered = scan - meanPoint
    return scan_centered

# Calculates M**(-1/2) for square matrix M
def sqrt_inv(M: np.ndarray) -> np.ndarray:
    if M.shape[0] != M.shape[1]:
        print("This matrix is not square")
    else:
        D, V = np.linalg.eig(M)
        out = np.matmul(V, np.matmul(np.diag(1.0 / np.sqrt(D)), V.T))
        return out

# Calculates the unbiased estimator of the std of a vector x, i.e.,
# we divide by m-1 instead of m
def std(x):
    m = len(x)
    return np.sqrt(np.var(x)*m/(m-1))

def var(x):
    m = len(x)
    return np.var(x)*m/(m-1)

# Calculates the covariance matrix Sigma_XZ between two data matrices X and Z
def cov_mat(X,Z):
    if X.shape[1] != Z.shape[1]:
        print("No covariance matrix possible")
    else:
        m = X.shape[1]
        Xc = center(X)
        Zc = center(Z)
```

```
        cov = (1/(m-1))*np.matmul(Xc,Zc.T)
        return cov

# Calculates the correlation matrix P_XZ between two data matrices X and Z
def cor_mat(X,Z):
    if X.shape[1] != Z.shape[1]:
        print("No correlation matrix possible")
    else:
        n = X.shape[0]
        p = Z.shape[0]
        cor = np.zeros((n,p))
        cov = cov_mat(X,Z)
        for i in range(n):
            for j in range(p):
                cor[i,j] = cov[i,j]/(std(X[i])*std(Z[j]))
        return cor

# Calculates the diagonal covariance matrix S_X of a matrix X
def diag_cov_mat(X):
    m = len(X[0])
    sigma = (m/(m-1))*X.var(axis=1)
    return np.diag(sigma)

# Shrinkage
# Calculates the shrunk covariance matrix of centered scan X and its covariance matrix cov
# (Schafer and Strimmer 2005)
def shrink_cov(X, cov):
    delta, T = shrink_intensity(X, cov)
    return delta*T + (1-delta)*cov

# Calculates the shrunk covariance matrix of scan X and its covariance matrix cov
# (Opgen-Rhein and Strimmer 2007)

# Calculates the optimal shrinkage intensity
def shrink_intensity(X, cov):
    n = X.shape[0]
    m = X.shape[1]
    w = np.zeros([n,n,m])
    w_bar = np.zeros([n,n])
    Var = np.zeros([n,n])
    var_sum = np.zeros(m)
    sum_cov = 0

    x_bar = X.mean(axis = 1)
    x_bar = x_bar[:, np.newaxis]

    for i in range(n):
        for j in range(n):
            if j !=i :
                sum_cov = sum_cov + (cov[i][j])**2
            for k in range(m):
                w[i][j][k] = (X[i][k]-x_bar[i])*(X[j][k]-x_bar[j])

            w_bar[i][j] = np.mean(w[i][j])

            for k in range(m):
                var_sum[k] = (w[i][j][k] - w_bar[i][j])**2
```

```
            if j !=i :
                Var[i][j] = (m/((m-1)**3))*sum(var_sum)

    T = diag_cov_mat(X)
    delta = (Var.sum())/(sum_cov)

    return delta*T + (1-delta)*cov

def Midpoint_Riemann_sum(x: np.array, y: np.array):
    delta_x = x[1] - x[0]
    Riemann_sum = y[0]*0.5*delta_x
    for i in range(1,len(x)-1):
        Riemann_sum = Riemann_sum + delta_x*y[i]

    Riemann_sum = Riemann_sum + y[-1]*0.5*delta_x
    return Riemann_sum
```

C.2. whitening_methods_functions

```
import numpy as np
from whitening_math_functions import *

# Different Whitening methods

# PCA whitening
def pca_whitening_matrix(X: np.ndarray, epsilon: np.float64) -> np.ndarray:
    cov = np.cov(X, rowvar=True)
    D, V = np.linalg.eigh(cov)
    out = np.matmul(np.diag(1.0 / np.sqrt(D + epsilon)), V.T)
    return out

def pca_on_cov(V: np.ndarray, D: np.ndarray, epsilon: np.float64) -> np.ndarray:
    out = np.matmul(np.diag(1.0 / np.sqrt(D + epsilon)), V.T)
    return out

# Standardized PCA
def pca_std_on_cov(X: np.ndarray, cov: np.ndarray, epsilon: np.float64) -> np.ndarray:
    P = np.corrcoef(X)
    Theta, G = np.linalg.eigh(P)
    m = len(X[0])
    S = (m/(m-1))*X.var(axis=1)
    out1 = np.matmul(sqrt_inv(np.diag(Theta + epsilon)), G.T)
    out2 = np.diag(1.0 / np.sqrt(S + epsilon))
    return np.matmul(out1, out2)

# ZCA whitening
def zca_whitening_matrix(X: np.ndarray, epsilon: np.float64) -> np.ndarray:
    cov = np.cov(X, rowvar=True)
    D, V = np.linalg.eigh(cov)
    out = np.matmul(V, np.matmul(np.diag(1.0 / np.sqrt(D + epsilon)), V.T))
    return out

def zca_on_cov(V: np.ndarray, D: np.ndarray, epsilon: np.float64) -> np.ndarray:
    out = np.matmul(V, np.matmul(np.diag(1.0 / np.sqrt(D + epsilon)), V.T))
    return out

# Standardized ZCA
def zca_std_on_cov(X: np.ndarray, cov: np.ndarray, epsilon: np.float64) -> np.ndarray:
    P = np.corrcoef(X)
    m = len(X[0])
    S = (m/(m-1))*X.var(axis=1)
    out = np.matmul(sqrt_inv(P + epsilon), np.diag(1.0 / np.sqrt(S + epsilon)))
    return out

# Cholesky whitening
def cholesky_on_cov(V: np.ndarray, D: np.ndarray, epsilon: np.float64) -> np.ndarray:
    cov_inv = np.matmul(V, np.matmul(np.diag(1.0 / (D + epsilon)), V.T))
    L = np.linalg.cholesky(cov_inv)
    return L.T

# CCA whitening
# CCA by eigendecomposition of H
def CCA_by_H(X: np.ndarray, Z: np.ndarray) -> np.ndarray:
    Sigma_X = np.cov(X)
```

```
    Sigma_Z = np.cov(Z)
    Sigma_XZ = cov_mat(X, Z)
    Sigma_ZX = Sigma_XZ.T
    DX, VX = np.linalg.eigh(Sigma_X)
    DZ, VZ = np.linalg.eigh(Sigma_Z)

    H1 = np.matmul(np.linalg.inv(Sigma_X), Sigma_XZ)
    H2 = np.matmul(np.linalg.inv(Sigma_Z), Sigma_ZX)
    H = np.matmul(H1, H2)
    for i in range(len(H)):
        for j in range(len(H[i])):
            if np.abs(H[i][j]) < 0.0000001:
                H[i][j] = 0
            if np.abs(1 - H[i][j]) < 0.0000001:
                H[i][j] = 1
    L, WX = np.linalg.eig(H)
    WZ = np.zeros([len(WX), len(WX)])
    for i in range(len(WX)):
        WZ[i] = np.matmul(np.matmul(np.linalg.inv(Sigma_Z), Sigma_ZX), WX[i])
    return WX, WZ
# cov_mat(np.matmul(WX[0],X),np.matmul(WX[1],X))

# CCA by singular value decomposition of K
def CCA_by_K(X: np.ndarray, Z: np.ndarray) -> np.ndarray:
    P_X = cor_mat(X, X)
    P_XZ = cor_mat(X, Z)
    P_Z = cor_mat(Z, Z)
    K = np.matmul(np.matmul(sqrt_inv(P_X), P_XZ), sqrt_inv(P_Z))
    Q_X, Lambda, Q_Z = np.linalg.svd(K)
    S_X = diag_cov_mat(X)
    S_Z = diag_cov_mat(Z)

    W_X = np.matmul(np.matmul(Q_X.T, sqrt_inv(P_X)), sqrt_inv(S_X))
    W_Z = np.matmul(np.matmul(Q_Z.T, sqrt_inv(P_Z)), sqrt_inv(S_Z))

    return W_X, W_Z

# CCA by sklearn

# n = np.linalg.matrix_rank(X)
# p = np.linalg.matrix_rank(Z)
# r = min(n,p)
# from sklearn.cross_decomposition import CCA
# cca = CCA(n_components=r)
# cca.fit(X, Z)
# X_cca, Z_cca = cca.transform(X, Z)

def Whitening(X: np.ndarray, method, observations, epsilon):
    if observations == 'pixels':
        X = X.T

    cov = np.cov(X, rowvar=True)
    centered = center(X)

    D, V = np.linalg.eigh(cov)
    for j in range(len(D)):
        if D[j] < 1e-6:
```

```
            D[j] = 0

    if method == 'ZCA':
        W = zca_on_cov(V, D, epsilon)
    if method == "Cholesky":
        W = cholesky_on_cov(V, D, epsilon)

    Y = np.matmul(W, centered)
    if observations == 'pixels':
        Y = Y.T
    return Y
```

C.3. whitening_SNR_functions

```
import numpy as np
import matplotlib.pyplot as plt
from whitening_math_functions import *
from whitening_methods_functions import *

#%%

# Create shot noise
def shot_noise(x: np.ndarray):
    shot_noise = np.zeros(548)
    for i in range(548):
        shot_noise[i] = np.random.normal(0, np.sqrt(x[i]))
    return x + shot_noise

def stack_shotnoise(x: np.ndarray, n: int):
    shotnoise = shot_noise(x)
    for i in range(1,n):
        shotnoise = np.vstack([shotnoise, shot_noise(x)])
    return shotnoise

# Fit baseline
def baseline_fit(x: np.array, y: np.array, degree):
    x_min_Gaussian = np.concatenate((x[:101], x[213:]))
    y_min_Gaussian = np.concatenate((y[:101], y[213:]))
    z = np.polyfit(x_min_Gaussian, y_min_Gaussian, deg = degree)
    baseline = np.zeros(len(x))
    for i in range(len(x)):
        baseline[i] = z[0]*(x[i]**6) + z[1]*(x[i]**5) + z[2]*(x[i]**4)
        + z[3]*(x[i]**3) + z[4]*(x[i]**2) + z[5]*x[i] + z[6]
    return baseline

def Ratio_signal_and_noise(Y: np.ndarray, target, x, MC: int, degree):
    target_pixel = Y[target]
    target_min_Gaussian = np.concatenate((target_pixel[:101], target_pixel[213:]))
    baseline = baseline_fit(x, target_pixel, degree)
    baseline_min_Gaussian = np.concatenate((baseline[:101], baseline[213:]))
    sigmaRMS = np.std(target_min_Gaussian - baseline_min_Gaussian)
    signal = Midpoint_Riemann_sum(x[120:194], target_pixel[120:194]- baseline[120:194])
    SNR = signal / sigmaRMS
    return SNR, signal, sigmaRMS

def plot_SNR(SNR, signal, noise, size, method, observations, MC: int):
    SNR_mean = SNR.mean(1)
    SNR_std = SNR.std(1)
    plt.plot(size, SNR_mean, color = 'r', label = f'Mean SNR of {MC} Monte Carlo simulations')
    plt.plot(size,SNR_mean + SNR_std, '--b', label = 'Standard deviation')
    plt.plot(size,SNR_mean - SNR_std, '--b')
    plt.title(f'Signal-to-Noise Ratio of {method} whitenend target spectrum \n \
                compared to total amount of whitened pixels. \n \
                    {observations} used as observations')
    plt.xlabel('Total number of whitened pixels')
    plt.ylabel('SNR')
    if method == "Cholesky" and observations == 'wavelengths':
        plt.legend(loc="upper left")
    else:
```

```
            plt.legend(loc="upper right")
        plt.show()
        plt.close()

        signal_mean = signal.mean(1)
        signal_std = signal.std(1)
        plt.plot(size, signal_mean, color = 'r', label = f'Mean SNR of {MC} Monte Carlo simulations')
        plt.plot(size,signal_mean + signal_std, '--b',  label = 'Standard deviation')
        plt.plot(size,signal_mean - signal_std, '--b')
        plt.title(f'Signal of {method} whitenend target spectrum \n \
                  compared to total amount of whitened pixels. \n \
                  {observations} used as observations')
        plt.xlabel('Total number of whitened pixels')
        plt.ylabel('Intensity')
        plt.legend(loc="upper right")
        plt.show()
        plt.close()

        noise_mean = noise.mean(1)
        noise_std = noise.std(1)
        plt.plot(size, noise_mean, color = 'r', label = f'Mean SNR of {MC} Monte Carlo simulations')
        plt.plot(size, noise_mean + noise_std, '--b',  label = 'Standard deviation')
        plt.plot(size, noise_mean - noise_std, '--b')
        plt.title(f'Noise of {method} whitenend target spectrum \n \
                  compared to total amount of whitened pixels. \n \
                  {observations} used as observations')
        plt.xlabel('Total number of whitened pixels')
        plt.ylabel('Intensity')
        plt.legend(loc="upper right")
        plt.show()
        plt.close()
        return
```

C.4. Whitening_synthetic_data

```
import sys
sys.path.append('../..')
sys.path.append("C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Jaar␣3/Thesis/kaleb/notebooks/Whitening_Gwyn")
import numpy as np
from kaleb.data import NestedDirectory
import seaborn as sns
import matplotlib.pyplot as plt
from whitening_SNR_functions import *
observations = "pixels"         # observations = "pixels" or "wavelengths"
noise = "with"                      # "with" or "without" shot noise
epsilon = 1e-4                      # epsilon = 0 or 1e-4

#%%

import pandas as pd
file_name = "C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/\
␣␣␣␣Jaar␣3/Thesis/Data/Teflon_met_Gaussian_function.xlsx"
sheet = 'Data'
df = pd.read_excel(io = file_name, sheet_name = sheet)
df = df.to_numpy(dtype = np.float64)

x = np.linspace(200, 550, 548)
y = df[0]
degree = 6
z = np.polyfit(x, y, deg = degree)
plt.plot(x,y, label='Average␣Teflon␣spectrum␣')
plt.plot(x, z[0]*(x**6) + z[1]*(x**5) + z[2]*(x**4) + z[3]*(x**3) + z[4]*(x**2) + z[5]*x + z[6],
         label=f'Fitted␣polynomial␣with␣degree␣{degree}')
plt.plot(x,df[3], 'y--', label='Spectrum␣target␣pixel')
plt.title('Average␣Teflon␣spectrum␣and␣fitted␣polynomial')
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.legend(loc="lower␣right", fancybox=True, shadow=True)
plt.show()
plt.close()

#%% # Make synthetic data without noise
def stack_teflon(n: int):
    teflon = df[1]
    for i in range(1,n):
        teflon = np.vstack([teflon, df[1]])
    return teflon

if noise == "without":
    teflon2 = np.vstack([df[1],df[1]])
    teflon3 = np.vstack([df[1],df[1],df[1]])

    syn12 = np.vstack([df[1],df[3]])
    syn22 = np.vstack([df[1],df[3],stack_teflon(2)])
    syn33 = np.vstack([stack_teflon(4),df[3],stack_teflon(4)])
    syn44 = np.vstack([stack_teflon(10),df[3],stack_teflon(5)])
    syn55 = np.vstack([stack_teflon(12),df[3],stack_teflon(12)])
    syn77 = np.vstack([stack_teflon(24),df[3],stack_teflon(24)])
    syn99 = np.vstack([stack_teflon(40),df[3],stack_teflon(40)])
```

```
    syn11 = np.vstack([stack_teflon(60),df[3],stack_teflon(60)])
    syn13 = np.vstack([stack_teflon(84),df[3],stack_teflon(84)])
    syn15 = np.vstack([stack_teflon(112),df[3],stack_teflon(112)])
    syn19 = np.vstack([stack_teflon(180),df[3],stack_teflon(180)])
    syn21 = np.vstack([stack_teflon(220),df[3],stack_teflon(220)])
    syn23 = np.vstack([stack_teflon(264),df[3],stack_teflon(264)])
    syn25 = np.vstack([stack_teflon(312),df[3],stack_teflon(312)])
    syn29 = np.vstack([stack_teflon(420),df[3],stack_teflon(420)])
    syn31 = np.vstack([stack_teflon(480),df[3],stack_teflon(480)])

#%% # Make synthetic data with noise

if noise == "with":
    syn12 = np.vstack([shot_noise(df[1]),shot_noise(df[3])])
    syn22 = np.vstack([shot_noise(df[1]),shot_noise(df[3]),stack_shotnoise(df[1], 2)])
    syn33 = np.vstack([stack_shotnoise(df[1], 4),shot_noise(df[3]),stack_shotnoise(df[1], 4)])
    syn44 = np.vstack([stack_shotnoise(df[1], 10),shot_noise(df[3]),stack_shotnoise(df[1], 5)])
    syn55 = np.vstack([stack_shotnoise(df[1], 12),shot_noise(df[3]),stack_shotnoise(df[1], 12)])
    syn77 = np.vstack([stack_shotnoise(df[1], 24),shot_noise(df[3]),stack_shotnoise(df[1], 24)])
    syn99 = np.vstack([stack_shotnoise(df[1], 40),shot_noise(df[3]),stack_shotnoise(df[1], 40)])

    syn11 = np.vstack([stack_shotnoise(df[1], 60),shot_noise(df[3]),stack_shotnoise(df[1], 60)])
    syn13 = np.vstack([stack_shotnoise(df[1], 84),shot_noise(df[3]),stack_shotnoise(df[1], 84)])
    syn15 = np.vstack([stack_shotnoise(df[1], 112),shot_noise(df[3]),stack_shotnoise(df[1], 112)])
    syn19 = np.vstack([stack_shotnoise(df[1], 180),shot_noise(df[3]),stack_shotnoise(df[1], 180)])
    syn21 = np.vstack([stack_shotnoise(df[1], 220),shot_noise(df[3]),stack_shotnoise(df[1], 220)])
    syn23 = np.vstack([stack_shotnoise(df[1], 264),shot_noise(df[3]),stack_shotnoise(df[1], 264)])
    syn25 = np.vstack([stack_shotnoise(df[1], 312),shot_noise(df[3]),stack_shotnoise(df[1], 312)])
    syn29 = np.vstack([stack_shotnoise(df[1], 420),shot_noise(df[3]),stack_shotnoise(df[1], 420)])

#%%

syn12_3d = syn12.reshape(1, 2, 548)
syn22_3d = syn22.reshape(2, 2, 548)
syn33_3d = syn33.reshape(3, 3, 548)
syn44_3d = syn44.reshape(4, 4, 548)
syn55_3d = syn55.reshape(5, 5, 548)
syn77_3d = syn77.reshape(7, 7, 548)
syn99_3d = syn99.reshape(9, 9, 548)

syn11_3d = syn11.reshape(11,11,548)
syn13_3d = syn13.reshape(13,13,548)
syn15_3d = syn15.reshape(15,15,548)
syn19_3d = syn19.reshape(19,19,548)
syn21_3d = syn21.reshape(21,21,548)
syn23_3d = syn23.reshape(23,23,548)
syn25_3d = syn25.reshape(25, 25, 548)
syn29_3d = syn29.reshape(29, 29, 548)

syn = [syn12,syn22,syn33,syn44,syn55,syn77,syn99,syn11,
       syn13,syn15,syn19,syn21,syn23,syn25,syn29]
syn_3d = [syn12_3d,syn22_3d,syn33_3d,syn44_3d,syn55_3d,
          syn77_3d,syn99_3d,syn11_3d,syn13_3d,syn15_3d,
          syn19_3d,syn21_3d,syn23_3d,syn25_3d,syn29_3d]
length = [1,2,3,4,5,7,9,11,13,15,19,21,23,25,29]
width = [2,2,3,4,5,7,9,11,13,15,19,21,23,25,29]
```

```
target = [1,1,4,10,12,24,40,60,84,112,180,220,264,312,420]      # index of the target pixel

#%%
# ZCA and Cholesky whitening

SNR_zca = np.zeros(len(syn))
SNR_chol = np.zeros(len(syn))
size = np.zeros(len(syn))

for i in range(len(syn)):
    size[i] = len(syn[i])

    if observations == 'pixels':
        syn[i] = syn[i].T

    cov = np.cov(syn[i], rowvar=True)
    centered = center(syn[i])
    D, V = np.linalg.eigh(cov)
    for j in range(len(D)):
        if D[j] < 1e-6:
            D[j] = 0

    W_zca = zca_on_cov(V, D, epsilon)

    Y_zca = np.matmul(W_zca,centered)
    if observations == 'pixels':
        Y_zca = Y_zca.T

    target_pixel_zca = Y_zca[target[i]]
    if noise == 'with':
        target_min_Gaussian_zca = np.concatenate((target_pixel_zca[:101], target_pixel_zca[213:]))
        baseline_zca = baseline_fit(x, target_pixel_zca, degree)
        baseline_min_Gaussian_zca = np.concatenate((baseline_zca[:101], baseline_zca[213:]))
        sigmaRMS_1 = np.std(target_pixel_zca[:101] - baseline_zca[:101])
        sigmaRMS_2 = np.std(target_pixel_zca[213:] - baseline_zca[213:])
        sigmaRMS_12_zca = np.std(target_min_Gaussian_zca - baseline_min_Gaussian_zca)
        integral_zca = Midpoint_Riemann_sum(x[120:194],
                                            target_pixel_zca[120:194]- baseline_zca[120:194])
        SNR_zca[i] = integral_zca / sigmaRMS_12_zca

    Y_zca_3d = Y_zca.reshape(length[i], width[i], 548)
    sns.heatmap(Y_zca_3d.sum(2), linewidths=1, linecolor='black')
    plt.title(f'ZCA Whitening, synthetic data {noise} shotnoise, \n \
              {observations} used as observations, epsilon = {epsilon}')
    plt.show()
    plt.close()

    plt.plot(x, Y_zca[0], label = 'Teflon spectrum')
    plt.plot(x, target_pixel_zca,label = 'Target spectrum', color = 'c')
    if noise == 'with':
        plt.plot(x, baseline_zca, label = 'Baseline of target spectrum', color = 'black')
    plt.axvline(x = 270, color = 'black')        # 300 - 3*sigma
    plt.axvline(x = 330, color = 'black')        # 300 + 3*sigma
    plt.legend(loc="lower right", fancybox=True, shadow=True)
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Intensity')
    plt.show()
```

```
    plt.close()

#%%
Z1 = Y_zca[0]
Z31 = Z1[117:196]
x3 = x[117:196]

# plot the main figure
plt.plot(x, Y_zca[0], label = 'Teflon spectrum')
plt.plot(x, target_pixel_zca,label = 'Target spectrum', color = 'c')
plt.legend(loc='lower right', ncol=1, fancybox=True, shadow=True)
plt.xlabel('Wavelength (nm)')
plt.ylabel('Intensity')

# location for the zoomed portion
sub_axes = plt.axes([.47, .3, .4, .2])
plt.yticks(visible=True)

# plot the zoomed portion
sub_axes.plot(x3, Z31, color = 'red')
plt.show()
plt.close()

#%%
Y_12 = Whitening(syn12, 'ZCA', 'wavelengths', epsilon)
plt.plot(x, Y_12[0], label = 'Teflon spectrum')
plt.plot(x, Y_12[1], label = 'Target spectrum', color = 'c')
plt.plot(x, Y_12[0] + Y_12[1], label = 'Sum of all spectra', color = 'orange')
plt.legend(loc="lower right", fancybox=True, shadow=True)
plt.xlabel('Wavelength (nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

#%%
Y_22 = Whitening(syn22, 'ZCA', 'wavelengths', epsilon)
plt.plot(x, Y_22[0], label = 'Teflon spectrum')
plt.plot(x, Y_22[1], label = 'Target spectrum', color = 'c')
plt.plot(x, Y_22[0] + Y_22[1] + Y_22[2] + Y_22[3],
         label = 'Sum of all spectra', color = 'orange')
plt.legend(loc="lower right", fancybox=True, shadow=True)
plt.xlabel('Wavelength (nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

#%%
Y_841 = Whitening(syn29, 'ZCA', 'wavelengths', epsilon)
plt.plot(x, Y_841[0], label = 'ZCA whitened Teflon spectrum')
#plt.plot(x, Y_841[420], label = 'Target spectrum', color = 'c')
sum_wav = np.zeros(548)
for i in range(841):
    sum_wav = sum_wav + Y_841[i]
plt.plot(x, (1/841)*sum_wav, label = 'Sum of all spectra in the scan',
         color = 'orange', linestyle = 'dashed')
plt.legend(loc="lower right", fancybox=True, shadow=True)
plt.xlabel('Wavelength (nm)')
```

```
plt.ylabel('Intensity')
plt.show()
plt.close()

#%%

Y_12 = Whitening(syn12, 'ZCA', 'wavelengths', epsilon)

polynomial = np.zeros(548)
p = np.zeros(548)
for j in range(548):
    polynomial[j] = z[0]*(x[j]**6) + z[1]*(x[j]**5) + z[2]*(x[j]**4) + z[3]*(x[j]**3)
    polynomial[j] = polynomial[j] + z[4]*(x[j]**2) + z[5]*x[j] + z[6]
sum_poly = polynomial.sum()
for k in range(548):
    p[k] = (polynomial[k] - sum_poly/548)
max_p = max(p)
max_sum = max(Y_12[0] + Y_12[1])
factor = max_sum / max_p
plt.plot(x, factor*p,
         label=f'Fitted polynomial with degree {degree}, normalised')
plt.plot(x, Y_12[0] + Y_12[1], label = 'Sum of all spectra', color = 'orange')
plt.legend(loc="lower right", fancybox=True, shadow=True)
plt.show()
plt.close()

#%%
av = (polynomial + df[3])/2
# av = polynomial
# av = df[3]
av_norm = np.zeros(548)
sum_av = av.sum()
for k in range(548):
    av_norm[k] = (av[k] - sum_av/548)
max_av_norm = max(av_norm)
max_sum = max(Y_12[0] + Y_12[1])
factor = max_sum / max_p
av_norm = factor*av_norm

def cos_similarity(P, Q):
    theta_up = np.sum(P*Q)
    theta_down = (np.sqrt(np.sum(P*P))*np.sqrt(np.sum(Q*Q)))
    return theta_up/theta_down
print(cos_similarity(av_norm, Y_12[0] + Y_12[1]))

#%%

Y_12_pix = Whitening(syn12, 'ZCA', 'pixels', epsilon)
Y_12_wav = Whitening(syn12, 'ZCA', 'wavelengths', epsilon)

target_pix = Y_12_pix[1]
target_wav = Y_12_wav[1]

target_min_Gaussian_pix = np.concatenate((target_pix[:101], target_pix[213:]))
baseline_pix = baseline_fit(x, target_pix, degree)
baseline_min_Gaussian_pix = np.concatenate((baseline_pix[:101], baseline_pix[213:]))
sigmaRMS_1 = np.std(target_pix[:101] - baseline_pix[:101])
```

```
sigmaRMS_2 = np.std(target_pix[213:] - baseline_pix[213:])
sigmaRMS_12_pix = np.std(target_min_Gaussian_pix - baseline_min_Gaussian_pix)
integral_pix = Midpoint_Riemann_sum(x[120:194], target_pix[120:194]- baseline_pix[120:194])
SNR_pix = integral_pix / sigmaRMS_12_pix

target_min_Gaussian_wav = np.concatenate((target_wav[:101], target_wav[213:]))
baseline_wav = baseline_fit(x, target_wav, degree)
baseline_min_Gaussian_wav = np.concatenate((baseline_wav[:101], baseline_wav[213:]))
sigmaRMS_1 = np.std(target_wav[:101] - baseline_wav[:101])
sigmaRMS_2 = np.std(target_wav[213:] - baseline_wav[213:])
sigmaRMS_12_wav = np.std(target_min_Gaussian_wav - baseline_min_Gaussian_wav)
integral_wav = Midpoint_Riemann_sum(x[120:194], target_wav[120:194]- baseline_wav[120:194])
SNR_wav = integral_wav / sigmaRMS_12_wav


plt.plot(x, Y_12_pix[0], label = 'Teflon␣spectrum')
plt.plot(x, target_pix,label = 'Target␣spectrum', color = 'c')
plt.plot(x, baseline_pix, label = 'Baseline␣of␣target␣spectrum', color = 'black')
plt.axvline(x = 270, color = 'black')       # 300 - 3*sigma
plt.axvline(x = 330, color = 'black')       # 300 + 3*sigma
plt.legend(loc="lower␣right", fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

plt.plot(x, Y_12_wav[0], label = 'Teflon␣spectrum')
plt.plot(x, target_wav,label = 'Target␣spectrum', color = 'c')
plt.plot(x, baseline_wav, label = 'Baseline␣of␣target␣spectrum', color = 'black')
plt.axvline(x = 270, color = 'black')       # 300 - 3*sigma
plt.axvline(x = 330, color = 'black')       # 300 + 3*sigma
plt.legend(loc="lower␣right", fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

#%%
plt.plot(x, target_pix - baseline_pix,label = 'Target␣spectrum␣minus␣baseline,\
␣␣␣␣␣␣␣␣␣pixels␣used␣as␣observations', color = 'c')
plt.legend(loc="upper␣left", fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()
plt.plot(x, target_wav - baseline_wav,label = 'Target␣spectrum␣minus␣baseline\
␣␣␣␣␣␣␣␣␣,␣wavelengths␣used␣as␣observations', color = 'c')
# plt.plot(x, baseline_wav, label = 'Baseline of target spectrum', color = 'black')
plt.legend(loc="upper␣left", fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()
print(cos_similarity(target_pix, target_wav - baseline_wav))

#%%
# CCA
```

```
# from defaults import Local
# from kaleb.data import read_config
# from kaleb.ml.similarity import Signatures

# algo_config = read_config(Local.ConfigSimilarity / 'ctc/ac3_3.yaml')
# signatures = Signatures.from_config(algo_config['targets'])
# an_signatures = signatures.targets_settings['AN']['data']
# petn_signatures = signatures.targets_settings['PETN']['data']
# rdx_signatures = signatures.targets_settings['RDX']['data']
# tnt_signatures = signatures.targets_settings['TNT']['data']
# Z = np.concatenate((an_signatures, petn_signatures, rdx_signatures, tnt_signatures))

# if observations == "wavelengths":
#     Wx_CCA, WzCCA = CCA_by_K(syn77,Z)
#     Yx = np.matmul(Wx_CCA, center(syn77))
#     Yz = np.matmul(Wz_CCA, center(Z))
#     Yx_3d = Yx.reshape(7, 7, 548)
#     sns.heatmap(Yx_3d.sum(2), linewidths=1, linecolor='black')
    # plt.title(f'CCA Whitening, synthetic data {noise} shotnoise, \n \
    #           {observations} used as observations')
#     plt.show()
#     plt.close()
#     Yz_3d = Yz.reshape(4, 4, 548)
#     sns.heatmap(Yz_3d.sum(2), linewidths=1, linecolor='black')
#     plt.title(f'CCA Whitening, signatures, \n {observations} used as observations')
#     plt.show()
#     plt.close()
```

C.5. SNR_Monte_Carlo

```
import sys
sys.path.append('../..')
sys.path.append("C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Jaar␣3/Thesis/kaleb/notebooks/Whitening_Gwyn")
import numpy as np
from kaleb.data import NestedDirectory
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from whitening_SNR_functions import *

epsilon = 1e-4                  # epsilon = 0 or 1e-4
MC = 1000                       # number of Monte Carlo simulations

#%%

import pandas as pd
file_name = "C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/\
␣␣␣␣Jaar␣3/Thesis/Data/Teflon_met_Gaussian_function.xlsx"
sheet = 'Data'
df = pd.read_excel(io = file_name, sheet_name = sheet)
df = df.to_numpy(dtype = np.float64)

x = np.linspace(200, 550, 548)
y = df[0]
degree = 6
z = np.polyfit(x, y, deg = degree)

#%%
def synthetic_scan(size: int):
    if size%2 == 0:
        t1 = int(size/2)
        t2 = t1 - 1
    else:
        t1 = int((size - 1)/2)
        t2 = t1
    scan = np.vstack([stack_shotnoise(df[1],t1),shot_noise(df[3]),stack_shotnoise(df[1],t2)])
    return scan, t1

def Monte_Carlo(size: np.ndarray, MC: int, pixels, wavelengths):
    arg = 0
    if max(size) >= 548:
        while size[arg] < 548:
            arg = arg + 1

    if pixels == 'yes':
        size_zca_pix = size
        SNR_zca_pix = np.zeros([len(size),MC])
        signal_zca_pix = np.zeros([len(size),MC])
        sigmaRMS_zca_pix = np.zeros([len(size),MC])
    if wavelengths == 'yes':
        size_zca_wav = size
        SNR_zca_wav = np.zeros([len(size),MC])
        signal_zca_wav = np.zeros([len(size),MC])
        sigmaRMS_zca_wav = np.zeros([len(size),MC])
```

```
    for m in range(MC):
        for i in range(len(size)):
            syn, target = synthetic_scan(size[i])

            if pixels == 'yes':
                syn_pix = syn.T
                cov_pix = np.cov(syn_pix, rowvar=True)
                centered_pix = center(syn_pix)
                D_pix, V_pix = np.linalg.eigh(cov_pix)
                for j in range(len(D_pix)):
                    if D_pix[j] < 1e-6:
                        D_pix[j] = 0
            if wavelengths == 'yes':
                cov_wav = np.cov(syn, rowvar=True)
                centered_wav = center(syn)
                D_wav, V_wav = np.linalg.eigh(cov_wav)
                for j in range(len(D_wav)):
                    if D_wav[j] < 1e-6:
                        D_wav[j] = 0

            if pixels == 'yes':
                W_zca_pix = zca_on_cov(V_pix, D_pix, epsilon)
                Y_zca_pix = np.matmul(W_zca_pix, centered_pix).T
                A = Ratio_signal_and_noise(Y_zca_pix, target, x, MC, degree)
                SNR_zca_pix[i][m], signal_zca_pix[i][m], sigmaRMS_zca_pix[i][m] = A
            if wavelengths == 'yes':
                W_zca_wav = zca_on_cov(V_wav, D_wav, epsilon)
                Y_zca_wav = np.matmul(W_zca_wav, centered_wav)
                A = Ratio_signal_and_noise(Y_zca_wav, target, x, MC, degree)
                SNR_zca_wav[i][m], signal_zca_wav[i][m], sigmaRMS_zca_wav[i][m] = A

    if pixels == 'yes':
        plot_SNR(SNR_zca_pix, signal_zca_pix, sigmaRMS_zca_pix, size_zca_pix,
                 'ZCA', 'pixels', MC)
    if wavelengths == 'yes':
        plot_SNR(SNR_zca_wav, signal_zca_wav, sigmaRMS_zca_wav, size_zca_wav,
                 'ZCA', 'wavelengths', MC)

    if pixels == 'yes' and wavelengths == 'no':
            return SNR_zca_pix, signal_zca_pix, sigmaRMS_zca_pix, size_zca_pix
    if pixels == 'no' and wavelengths == 'yes':
            return SNR_zca_wav, signal_zca_wav, sigmaRMS_zca_wav, size_zca_wav
    if pixels == 'yes' and wavelengths == 'yes':
            return SNR_zca_pix, size_zca_pix, SNR_zca_wav, size_zca_wav
    return

#%%
# Monte Carlo

# length = [3,5, 5, 5, 5, 5, 5, 5,10, 9,10,11,12,14,20,18,20,22,20,20,20,30,25]
# width = [3, 5,10,15,20,25,30,35,20,25,25,25,25,25,20,25,25,25,30,35,40,30,40]
size = [9,25,50,75,100,125,150,175,200,225,250,275,300,350,400,450,500,550,600,700,800,900,1000]

SNR_zca_pix, size_zca_pix, SNR_zca_wav, size_zca_wav = Monte_Carlo(size, MC, 'yes', 'yes')

#%%
```

```
np.savetxt(f"SNR_ZCA_whitening_{MC}x23_obs=pix.csv", SNR_zca_pix, delimiter=',␣')
np.savetxt(f"SNR_ZCA_whitening_{MC}x23_obs=wav.csv", SNR_zca_wav, delimiter=',␣')
```

C.6. Multi_area_whitening

```
import sys
sys.path.append('../..')
sys.path.append("C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Jaar␣3/Thesis/kaleb/notebooks/Whitening_Gwyn")
import numpy as np
from kaleb.data import NestedDirectory
import seaborn as sns
import matplotlib.pyplot as plt
from whitening_SNR_functions import *

epsilon = 1e-4                    # epsilon = 0 or 1e-4
MC = 1                          # number of Monte Carlo simulations

#%%

import pandas as pd
file_name = "C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/Jaar␣3/\
␣␣␣␣Thesis/Data/Teflon_met_Gaussian_function.xlsx"
sheet = 'Data'
df = pd.read_excel(io = file_name, sheet_name = sheet)
df = df.to_numpy(dtype = np.float64)

x = np.linspace(200, 550, 548)
y = df[0]
degree = 6
z = np.polyfit(x, y, deg = degree)
Gauss = -df[2]
Gauss = (1/1000)*Gauss
plt.plot(x,Gauss)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

#%%
def synthetic_scan(size: int):
    if size%2 == 0:
        t1 = int(size/2)
        t2 = t1 - 1
    else:
        t1 = int((size - 1)/2)
        t2 = t1
    scan = np.vstack([stack_shotnoise(df[1], t1),shot_noise(df[3]),stack_shotnoise(df[1], t2)])
    return scan, target1

# Multi-area whitening
def Leather_spectrum(n: int, x):
    y_data = np.zeros(len(x))
    for i in range(len(x)):
        if i < n:
            y_data[i] = 3400 + 46600/(1+np.exp(-((1/10))*(x[i] - (x[n] -100) )))
        if i >= n:
            y_data[i] = 50000 - 46600/(1+np.exp(-(1/10)*(x[i] - (x[n] + 100) )))
    return y_data
```

```
def MAW(y_data, total_pixels, leather_pixels, observations, method):
    teflon_pixels = total_pixels - leather_pixels
    syn_1, target_index = synthetic_scan(teflon_pixels)
    syn_2 = stack_shotnoise(y_data, leather_pixels)
    syn = np.vstack([syn_1, syn_2])
    if observations == 'pixels':
        syn = syn.T
        syn_1 = syn_1.T
        syn_2 = syn_2.T

    cov_1 = np.cov(syn_1, rowvar=True)
    cov_2 = np.cov(syn_2, rowvar=True)
    cov = np.cov(syn, rowvar=True)
    centered_1 = center(syn_1)
    centered_2 = center(syn_2)
    centered = center(syn)

    D, V = np.linalg.eigh(cov)
    D1, V1 = np.linalg.eigh(cov_1)
    D2, V2 = np.linalg.eigh(cov_2)
    for j in range(len(D)):
        if D[j] < 1e-6:
            D[j] = 0
    for j in range(len(D1)):
        if D1[j] < 1e-6:
            D1[j] = 0
    for j in range(len(D2)):
        if D2[j] < 1e-6:
            D2[j] = 0

    if method == 'ZCA':
        W = zca_on_cov(V, D, epsilon)
        W_1 = zca_on_cov(V1, D1, epsilon)
        W_2 = zca_on_cov(V2, D2, epsilon)
    if method == 'Cholesky':
        W = cholesky_on_cov(V, D, epsilon)
        W_1 = cholesky_on_cov(V1, D1, epsilon)
        W_2 = cholesky_on_cov(V2, D2, epsilon)


    Y_1 = np.matmul(W_1,centered_1)
    Y_2 = np.matmul(W_2,centered_2)
    Y = np.matmul(W,centered)
    if observations == 'pixels':
        Y = Y.T
        Y_1 = Y_1.T
        Y_2 = Y_2.T

    Y_maw = np.vstack([Y_1, Y_2])

    return Y, Y_maw, target_index

def plot_MAW(y_data, total_pixels, leather_pixels, observations, method):
    Y_zca, Y_zca_maw, target = MAW(y_data, total_pixels, leather_pixels, observations, method)
    plt.plot(x, Y_zca[target], label = f'Global {method} whitening')
    plt.plot(x, Y_zca_maw[target], label = f'Multi-area {method} whitening')
    # plt.title(f'Spectrum of target pixel using global and multi-area {method} whitening',\
```

```
    #           \n {observations} used as observations')
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Intensity')
    plt.legend(loc="lower right")
    plt.show()
    plt.close()
    return Y_zca, Y_zca_maw, target

def MAW_MC(size: np.ndarray, total_pixels: int, MC: int, pixels, wavelengths, method):
    if pixels == 'yes':
        SNR_pix = np.zeros([len(size),MC])
        signal_pix = np.zeros([len(size),MC])
        sigmaRMS_pix = np.zeros([len(size),MC])
        SNR_pix_maw = np.zeros([len(size),MC])
        signal_pix_maw = np.zeros([len(size),MC])
        sigmaRMS_pix_maw = np.zeros([len(size),MC])
    if wavelengths == 'yes':
        SNR_wav = np.zeros([len(size),MC])
        signal_wav = np.zeros([len(size),MC])
        sigmaRMS_wav = np.zeros([len(size),MC])
        SNR_wav_maw = np.zeros([len(size),MC])
        signal_wav_maw = np.zeros([len(size),MC])
        sigmaRMS_wav_maw = np.zeros([len(size),MC])

    leather_spectrum = Leather_spectrum(400, x)

    for m in range(MC):
        for i in range(len(size)):
            leather_pixels = size[i]
            if pixels == 'yes':
                A = MAW(leather_spectrum, total_pixels, leather_pixels, 'pixels', method)
                Y_pix, Y_pix_maw, target = A
                B = Ratio_signal_and_noise(Y_pix, target, x, MC, degree)
                SNR_pix[i][m], signal_pix[i][m], sigmaRMS_pix[i][m] = B
                C = Ratio_signal_and_noise(Y_pix_maw, target, x, MC, degree)
                SNR_pix_maw[i][m], signal_pix_maw[i][m], sigmaRMS_pix_maw[i][m] = C
            if wavelengths == 'yes':
                A = MAW(leather_spectrum, total_pixels, leather_pixels, 'wavelengths', method)
                Y_wav, Y_wav_maw, target = A
                B = Ratio_signal_and_noise(Y_wav, target, x, MC, degree)
                SNR_wav[i][m], signal_wav[i][m], sigmaRMS_wav[i][m] = B
                C = Ratio_signal_and_noise(Y_wav_maw, target, x, MC, degree)
                SNR_wav_maw[i][m], signal_wav_maw[i][m], sigmaRMS_wav_maw[i][m] = C

    if pixels == 'yes':
        plot_SNR(SNR_pix, signal_pix, sigmaRMS_pix, size, method, 'pixels', MC)
        plot_SNR(SNR_pix_maw, signal_pix_maw, sigmaRMS_pix_maw, size, method, 'pixels', MC)
    if wavelengths == 'yes':
        plot_SNR(SNR_wav, signal_wav, sigmaRMS_wav, size, method, 'wavelengths', MC)
        plot_SNR(SNR_wav_maw, signal_wav_maw, sigmaRMS_wav_maw, size, method, 'wavelengths', MC)

    if pixels == 'yes' and wavelengths == 'no':
        return SNR_pix, signal_pix, sigmaRMS_pix, SNR_pix_maw, signal_pix_maw, sigmaRMS_pix_maw
    if pixels == 'no' and wavelengths == 'yes':
        return SNR_wav, signal_wav, sigmaRMS_wav, SNR_wav_maw, signal_wav_maw, sigmaRMS_wav_maw
    if pixels == 'yes' and wavelengths == 'yes':
        return SNR_pix, SNR_wav, SNR_pix_maw, SNR_wav_maw
```

```
    return

#%%
# Scan of 200 pixels consisting of a varying number of leather pixels
# and the rest is teflon pixels with one target pixel.
leather_pixels = [2,3,4,5,6,7,8,9,10,15,20,25,50,75,100,125,150,175]
total_pixels = 200
SNR = MAW_MC(leather_pixels, total_pixels, MC, 'yes', 'yes')
SNR_pix, SNR_wav, SNR_pix_maw, SNR_wav_maw = SNR

#%%
# Synthetic data, different to teflon
# y_data1 = np.zeros(len(x))
# for i in range(len(x)):
#     if i < 391:
#         y_data1[i] = 3400 + 600/(1+np.exp(-((1/10))*(x[i] - 350)))
#     if i >= 391:
#         y_data1[i] = 4000 - 600/(1+np.exp(-(1/10)*(x[i]-550)))
# plt.plot(x,y_data1)

observations = 'wavelengths'
# plot_MAW(y_data1,42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(500, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(475, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(450, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(425, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(400, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(375, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(350, x),42,21,observations, 'ZCA')
plot_MAW(Leather_spectrum(325, x),42,21,observations, 'ZCA')

#%%
leather_spectrum = Leather_spectrum(400, x)
Y, Y_maw, target_index = MAW(leather_spectrum, 42, 21, 'pixels', 'ZCA')
plt.plot(x, Y[target_index], label = 'Target␣spectrum␣after␣global␣ZCA␣whitening')
plt.plot(x, Y_maw[target_index], label = 'Target␣spectrum␣after␣multi-area␣ZCA␣whitening')
plt.plot(x, baseline_fit(x, Y[target_index], 6), label = 'Fitted␣baseline␣to␣global␣\
␣␣␣␣␣␣␣␣ZCA␣whitened␣\n␣target␣spectrum', color = 'orange')
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.legend(loc = 'lower␣right', ncol=1, fancybox=True, shadow=True)
plt.show()
plt.close()

#%%
# Spectral Angle Mapper, returns a value between 0 and pi, lower = higher similarity
# 0 : P = Q
def SAM(P, Q):
    theta_up = np.sum(P*Q)
    theta_down = (np.sqrt(np.sum(P*P))*np.sqrt(np.sum(Q*Q)))
    theta = np.arccos(theta_up / theta_down)
    return theta

# Cosine similarity, returns a value between -1 and 1,
#-1: P = -Q,
# 1: P = Q,
# 0: P and Q orthogonal, P = 0 when Q != 0 and P != 0 when Q = 0
```

```
def cos_similarity(P, Q):
    theta_up = np.sum(P*Q)
    theta_down = (np.sqrt(np.sum(P*P))*np.sqrt(np.sum(Q*Q)))
    return theta_up/theta_down

#%%
points = np.linspace(300,500,9, dtype = int)
cossim_pix = np.zeros(len(points))
cossim_wav = np.zeros(len(points))
for i in range(len(points)):
    A = plot_MAW(Leather_spectrum(points[i], x), 200, 100, 'pixels', 'ZCA')
    Y_zca_pix, Y_zca_maw_pix, target_index_pix = A
    B = plot_MAW(Leather_spectrum(points[i], x), 200, 100, 'wavelengths', 'ZCA')
    Y_zca_wav, Y_zca_maw_wav, target_index_wav = B
    cossim_pix[i] = cos_similarity(Y_zca_pix[target_index_pix], Y_zca_maw_pix[target_index_pix])
    cossim_wav[i] = cos_similarity(Y_zca_wav[target_index_wav], Y_zca_maw_wav[target_index_wav])
    print(points[i], cossim_pix[i], cossim_wav[i])

#%%

def MAW_cos_similarity(leather_pixels, MC, epsilon, method, size):
    cos_sim_pix = np.zeros([len(leather_pixels),MC])
    cos_sim_maw_pix = np.zeros([len(leather_pixels),MC])
    cos_sim_wav = np.zeros([len(leather_pixels),MC])
    cos_sim_maw_wav = np.zeros([len(leather_pixels),MC])
    x = np.linspace(200, 550, 548)

    for j in range(len(leather_pixels)):
        for m in range(MC):
            A = MAW(Leather_spectrum(400, x), size, leather_pixels[j], 'pixels', method)
            Y_zca_pix, Y_zca_maw_pix, target_index_pix = A
            B = MAW(Leather_spectrum(400, x), size, leather_pixels[j], 'wavelengths', method)
            Y_zca_wav, Y_zca_maw_wav, target_index_wav = B
            cos_sim_pix[j][m] = cos_similarity(Y_zca_pix[target_index_pix], Gauss)
            cos_sim_maw_pix[j][m] = cos_similarity(Y_zca_maw_pix[target_index_pix], Gauss)
            cos_sim_wav[j][m] = cos_similarity(Y_zca_wav[target_index_wav], Gauss)
            cos_sim_maw_wav[j][m] = cos_similarity(Y_zca_maw_wav[target_index_wav], Gauss)

    plt.plot(leather_pixels, cos_sim_pix.mean(1), marker='o', linestyle='None',
             markersize=8, label = f'Global {method} whitening', color = 'red')
    plt.plot(leather_pixels, cos_sim_pix.mean(1) + cos_sim_pix.std(1), marker='o',
             linestyle='None', markersize=8, label = f'Global {method} whitening \
+- standard deviation', color = 'orange')
    plt.plot(leather_pixels, cos_sim_pix.mean(1) - cos_sim_pix.std(1), marker='o',
             linestyle='None', markersize=8, color = 'orange')
    plt.plot(leather_pixels, cos_sim_maw_pix.mean(1), marker='o', linestyle='None',
             markersize=8, label = f'Multi-area {method} whitening', color = 'blue')
    plt.plot(leather_pixels, cos_sim_maw_pix.mean(1) + cos_sim_maw_pix.std(1),
             marker='o', linestyle='None', markersize=8, label = f'Multi-area \
{method} whitening +- standard deviation', color = 'c')
    plt.plot(leather_pixels, cos_sim_maw_pix.mean(1) - cos_sim_maw_pix.std(1),
             marker='o', linestyle='None', markersize=8, color = 'c')
    # plt.title('Cosine similarty of target spectrum from a scan of 200 pixels,\
    #           \n where the number of leather pixels is displayed on the x-axis, \
    #               \n pixels used as observations.')
    plt.xlabel('Number of leather pixels of total 1000 pixels')
    plt.ylabel('Cosine similarity')
```

```
    plt.legend(loc='lower␣left', ncol=1, fancybox=True, shadow=True)
    plt.show()
    plt.close()
    # Target spectrum of global ZCA compared to that of a target pixel
    # on a 200 pixel teflon scan
    # Target spectrum of multi-area ZCA compared to that of a target pixel
    # on a scan with the same amount of teflon pixels
    plt.plot(leather_pixels, cos_sim_wav.mean(1), marker='o', linestyle='None',
             markersize=8, label = f'Global␣{method}␣whitening', color = 'red')
    plt.plot(leather_pixels, cos_sim_wav.mean(1) + cos_sim_wav.std(1), marker='o',
             linestyle='None', markersize=8, label = f'Global␣{method}␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣whitening␣+-␣standard␣deviation', color = 'orange')
    plt.plot(leather_pixels, cos_sim_wav.mean(1) - cos_sim_wav.std(1), marker='o',
             linestyle='None', markersize=8, color = 'orange')
    plt.plot(leather_pixels, cos_sim_maw_wav.mean(1), marker='o', linestyle='None',
             markersize=8, label = f'Multi-area␣{method}␣whitening', color = 'blue')
    plt.plot(leather_pixels, cos_sim_maw_wav.mean(1) + cos_sim_maw_wav.std(1),
             marker='o', linestyle='None', markersize=8, label = f'Multi-area␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{method}␣whitening␣+-␣standard␣deviation', color = 'c')
    plt.plot(leather_pixels, cos_sim_maw_wav.mean(1) - cos_sim_maw_wav.std(1),
             marker='o', linestyle='None', markersize=8, color = 'c')
    # plt.title('Cosine similarty of target spectrum from a scan of 200 pixels,\
    #           \n where the number of leather pixels is displayed on the x-axis,\
    #               \n wavelengths used as observations.')
    # plt.xlabel('Number of leather pixels of total 200 pixels')
    plt.ylabel('Cosine␣similarity')
    plt.legend(loc='lower␣left', ncol=1, fancybox=True, shadow=True)
    plt.show()
    plt.close()
    return cos_sim_pix, cos_sim_maw_pix,  cos_sim_wav, cos_sim_maw_wav

leather_pixels = [2,3,4,5,6,7,8,9,10,15,20,25,50,75,100,125,150,175]
cos_sim = MAW_cos_similarity(leather_pixels, 1, epsilon, 'ZCA', 200)
cos_sim_pix, cos_sim_maw_pix, cos_sim_wav, cos_sim_maw_wav = cos_sim
```

C.7. Normal_test

```
import sys
sys.path.append('../..')
sys.path.append("C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/Jaar␣3/\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Thesis/kaleb/notebooks/Whitening_Gwyn")
import numpy as np
import matplotlib.pyplot as plt
import csv
from scipy import stats
with open('SNR␣data/ZCA/SNR/SNR_ZCA_whitening_1000x23_obs=wav.csv', newline='') as csvfile:
    SNR_zca = list(csv.reader(csvfile))
    SNR_zca = np.array(SNR_zca, dtype = float)

import scipy
test = scipy.stats.normaltest(SNR_zca, axis=1, nan_policy='propagate')

import statsmodels.api as sm

size = [9,25,50,75,100,125,150,175,200,225,250,275,300,350,400,450,500,550,600,700,800,900,1000]
for i in range(len(size)):
    plt.hist(SNR_zca[i])
    plt.title(f'{size[i]}')
    plt.show()
    plt.close()

#%%
mean_SNR_zca = SNR_zca.mean(1)
std_SNR_zca = SNR_zca.std(1)
alpha = 0.05
from scipy.stats import kstest

stat_ks = np.zeros(len(size))
p_ks = np.zeros(len(size))
for i in range(len(size)):
    normed_data=(SNR_zca[i]-mean_SNR_zca[i])/std_SNR_zca[i]
    stat_ks[i], p_ks[i] = kstest(normed_data, stats.norm.cdf)
    print(size[i], stat_ks[i], p_ks[i])
    if p_ks[i] < 0.05:
        print('We␣reject␣the␣hypothesis␣that␣the␣data␣comes␣from␣a␣normal␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣distibution␣with␣mean␣', mean_SNR_zca[i], '␣and␣standard␣deviation␣',
              std_SNR_zca[i], '␣for␣a␣scan␣of␣', size[i], '␣pixels')
```

C.8. Whitening_Figures

```
import sys
sys.path.append('../..')
sys.path.append("C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/Jaar␣3/\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Thesis/kaleb/notebooks/Whitening_Gwyn")
import numpy as np
from kaleb.data import NestedDirectory
import seaborn as sns
import matplotlib.pyplot as plt

#%%
import pandas as pd
file_name = "C:/Users/Gwynn/OneDrive/Documenten/TW␣Studie/Master/Jaar␣3/\
␣␣␣␣Thesis/Data/Teflon_met_Gaussian_function.xlsx"
sheet = 'Data'
df = pd.read_excel(io = file_name, sheet_name = sheet)
df = df.to_numpy(dtype = np.float64)

x = np.linspace(200, 550, 548)
y = df[0]
degree = 6
z = np.polyfit(x, y, deg = degree)
plt.plot(x, y, label = 'Average␣Teflon␣spectrum')
plt.plot(x, z[0]*(x**6) + z[1]*(x**5) + z[2]*(x**4) + z[3]*(x**3) + z[4]*(x**2) + z[5]*x + z[6],
         label = 'Fitted␣polynomial␣with␣degree␣6', color = 'blue')
plt.legend(loc='upper␣left', ncol=1, fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')
plt.show()
plt.close()

Z1 = z[0]*(x**6) + z[1]*(x**5) + z[2]*(x**4) + z[3]*(x**3) + z[4]*(x**2) + z[5]*x + z[6]
Z2 = df[3]
Z31 = Z1[117:196]
Z32 = Z2[117:196]
x3 = x[117:196]

# plot the main figure
plt.plot(x,Z1, label = 'Fitted␣polynomial␣with␣degree␣6', color = 'blue')
plt.plot(x, Z2, label = 'Spectrum␣target␣substance␣on␣Teflon', color = 'orange',
         linestyle='dashed')
plt.legend(loc='upper␣left', ncol=1, fancybox=True, shadow=True)
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Intensity')

# location for the zoomed portion
sub_axes = plt.axes([.55, .2, .3, .3])
plt.yticks(visible=True)

# plot the zoomed portion
sub_axes.plot(x3, Z31, color = 'blue')
sub_axes.plot(x3, Z32, color = 'orange', linestyle='dashed')

# insert the zoomed figure
# plt.setp(sub_axes)
plt.show()
```

```
plt.close()

#%%
import csv
with open('SNR data/ZCA/SNR/SNR_ZCA_whitening_1000x23_obs=wav.csv', newline='') as csvfile:
    SNR_zca = list(csv.reader(csvfile))
    SNR_zca = np.array(SNR_zca, dtype = float)

mean_SNR_zca = SNR_zca.mean(1)
std_SNR_zca = SNR_zca.std(1)
size_6 = [550,600,700,800,900,1000]
size_9 = [400,450,500,550,600,700,800,900,1000]
size_14 = [9,25,50,75,100,125,150,175,200,225,250,275,300,350]
size_17 = [9,25,50,75,100,125,150,175,200,225,250,275,300,350,400,450,500]
size_21 = [530,535,540,541,542,543,544,545,546,547,548,549,551,552,553,554,555,556,557,558,559]
size_23 = [9,25,50,75,100,125,150,175,200,225,250,275,300,350,400,450,500,550,600,700,800,900,
           1000]
size_52 = [155,160,165,170,180,185,190,195,205,210,215,220,230,235,240,245,255,260,265,270,280,
        285,290,295,305,310,315,320,325,330,335,340,345,355,360,365,370,380,385,390,395,405,
        410,415,420,425,430,435,440,445,475,525]
size_75 = [9,25,50,75,100,125,150,155,160,165,170,175,180,185,190,195,200,205,210,215,220,225,
          230,235,240,245,250,255,260,265,270,275,280,285,290,295,300,305,310,315,320,325,330,
          335,340,345,350,355,360,365,370,380,385,390,395,400,405,410,415,420,425,430,435,440,
          445,450,475,500,525,550,600,700,800,900,1000]
size_85 = [9,25,50,75,100,125,150,155,160,165,170,175,180,185,190,195,200,205,210,215,220,225,
          230,235,240,245,250,255,260,265,270,275,280,285,290,295,300,305,310,315,320,325,330,
          335,340,345,350,355,360,365,370,380,385,390,395,400,405,410,415,420,425,430,435,440,
          445,450,475,500,525,550,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,
          1800,1900,2000]
if len(SNR_zca) == 6:
    size = size_6
if len(SNR_zca) == 9:
    size = size_9
if len(SNR_zca) == 14:
    size = size_14
if len(SNR_zca) == 17:
    size = size_17
if len(SNR_zca) == 21:
    size = size_21
if len(SNR_zca) == 23:
    size = size_23
if len(SNR_zca) == 75:
    size = size_75
if len(SNR_zca) == 85:
    size = size_85
plt.plot(size, mean_SNR_zca, color='red', marker='o', linestyle='None', markersize=8,
         label='Signal-to-noise ratio of target pixel')
plt.plot(size, mean_SNR_zca-std_SNR_zca, color='blue', marker='o', linestyle='None',
         markersize=8, label='Signal-to-noise ratio + standard deviation')
plt.plot(size, mean_SNR_zca+std_SNR_zca, color='blue', marker='o', linestyle='None',
         markersize=8)
plt.xlabel('Number of whitened pixels')
plt.ylabel('SNR')
plt.legend(loc='upper left', ncol=1, fancybox=True, shadow=True) # bbox_to_anchor=(1.0, 1.015)
plt.show()
plt.close()
```

```
#%%
def Leather_spectrum(n: int, x):
    y_data = np.zeros(len(x))
    for i in range(len(x)):
        if i < n:
            y_data[i] = 3400 + 46600/(1+np.exp(-((1/10))*(x[i] - (x[n] -100) )))
        if i >= n:
            y_data[i] = 50000 - 46600/(1+np.exp(-(1/10)*(x[i] - (x[n] + 100) )))
    return y_data

plt.plot(x, Leather_spectrum(500, x), label = "500")
plt.plot(x, Leather_spectrum(475, x), label = "475")
plt.plot(x, Leather_spectrum(450, x), label = "450")
plt.plot(x, Leather_spectrum(425, x), label = "425")
plt.plot(x, Leather_spectrum(400, x), label = "400")
plt.plot(x, Leather_spectrum(375, x), label = "375")
plt.plot(x, Leather_spectrum(350, x), label = "350")
plt.plot(x, Leather_spectrum(325, x), label = "325")
plt.legend(loc='upper left', ncol=1, fancybox=True, shadow=True)
plt.show()
plt.close()

#%%
import csv
with open('Multi-area-whitening/CosineSimilarity_Global\
          _ZCA_1000x18_obs=wav.csv', newline='') as csvfile:
    global_zca = list(csv.reader(csvfile))
    global_zca = np.array(global_zca, dtype = float)

mean_global_zca = global_zca.mean(1)
std_global_zca = global_zca.std(1)

with open('Multi-area-whitening/CosineSimilarity_Multi_Area\
          _ZCA_1000x18_obs=wav.csv', newline='') as csvfile:
    ma_zca = list(csv.reader(csvfile))
    ma_zca = np.array(ma_zca, dtype = float)

mean_ma_zca = ma_zca.mean(1)
std_ma_zca = ma_zca.std(1)

leather_pixels = [2,3,4,5,6,7,8,9,10,15,20,25,50,75, 100,125,150,175]
plt.errorbar(leather_pixels, mean_global_zca, std_global_zca,  marker='o', linestyle='None',
             markersize=8, ecolor = 'orange', elinewidth = 2, capsize = 5, capthick = 2,
             label = 'Global ZCA whitening + - standard deviation', color = 'red')
plt.xlabel('Number of leather pixels of total 200 pixels')
plt.ylabel('Cosine similarity')
plt.legend(loc='lower left', ncol=1, fancybox=True, shadow=True)
plt.show()
plt.close()
plt.errorbar(leather_pixels, mean_ma_zca, std_ma_zca, marker='o', linestyle='None',
             markersize=8, ecolor = 'c', elinewidth = 2, capsize = 5, capthick = 2,
             label = 'Multi-area ZCA whitening + - standard deviation', color = 'blue')
plt.xlabel('Number of leather pixels of total 200 pixels')
plt.ylabel('Cosine similarity')
plt.ylim(0.85,0.95)
plt.legend(loc='lower left', ncol=1, fancybox=True, shadow=True)
plt.show()
```

```
plt.close()
```