UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Study of machine learning techniques for image compression

BACHELOR FINAL THESIS

Document:
Report

Author:
Francesc Antich Navarro

Director /Co-director:
Alex Ferrer Ferre

Degree:
Bachelor's degree in Aerospace Vehicle Engineering

Examination session:
Spring, 2022

Mathematicians have various ways of judging the merits of new theorems and constructions. One very important criterion is esthetic — some developments just "feel" right, fitting, and beautiful.

— Ingrid Daubechies, *Wavelets and Other Phase Space Localization Methods*

Acknowledgements

I would like to thank my tutor, Alex Ferrer, for all his support and dedication during this last year motivating me to continue learning. Likewise, I would also like to thank my family, who have dedicated all their time, effort and resources to educate me and train me the best possible to face my future. Finally, I would like to dedicate this thesis to my godmother Joana, who has taught me to never stop seeing the world with illusion and joy.

Agraïments

M'agradaria agrair al meu tutor, Alex Ferrer, per tot el seu suport i dedicació durant aquest últim any motivant-me a continuar aprenent. Així mateix, també vull agrair a la meva família, que m'ha dedicat tot el seu temps, esforç i recursos per educar-me i formar-me el millor possible per afrontar la vida. Finalment, m'agradaria dedicar aquest treball a la meva padrina Joana qui m'ha ensenyat a mai deixar de veure el món des de la il·lusió i alegria.

Abstract

In the age of the Internet and cloud-based applications, image compression has become increasingly important. Moreover, image processing has recently sparked the interest of technology companies as autonomous machines powered by artificial intelligence using images as input are rapidly growing. Reducing the amount of information needed to represent an image is key to reducing the amount of storage space, transmission bandwidth, and computation time required to process the image, which in turn saves resources, energy, and money.

This study aims to investigate machine learning techniques (Fourier, wavelets, and PCA) for image compression. Several Fourier and wavelet methods are included, such as the well-known Cooley-Tukey algorithm, the discrete cosine transform, and the Mallart algorithm, among others. To comprehend each step of image compression, an object-oriented Matlab code has been developed in-house. To do so, extensive research in machine learning techniques, not only in terms of theoretical understanding, but also in the mathematics that support it.

The developed code is used to compare the performance of the different compression techniques studied. The findings of this study are consistent with the advances in image compression technologies in recent years, with the dominance of the JPEG compression method (Fourier) and later JPEG2000 (wavelets) reigning supreme.

Resum

La compressió d'imatges cada vegada és més important a l'era d'Internet i les aplicacions basades en el núvol. A més, el processament d'imatges ha despertat recentment l'interès de les empreses tecnològiques a causa del ràpid creixement de les màquines autònomes dotades d'intel·ligència artificial que utilitzen imatges com a entrada. La reducció de la quantitat d'informació necessària per representar una imatge és clau per reduir la quantitat d'espai d'emmagatzematge necessari, l'ample de banda de transmissió i el temps de computació requerit per processar la imatge, el que al seu torn estalvia recursos, energia i diners.

Aquest estudi pretén estudiar les tècniques d'aprenentatge automàtic (Fourier, Wavelets i PCA) per a la compressió d'imatges. S'inclouen diversos mètodes de Fourier i wavelets, com el conegut algorisme de Cooley-Tukey, la transformada discreta del cosinus i l'algoritme de Mallart, entre d'altres. Per comprendre cada pas de la compressió d'imatges, s'ha desenvolupat internament un codi Matlab orientat a objectes. Per això, s'ha dut a terme una àmplia investigació en tècniques d'aprenentatge automàtic, no només pel que fa a la comprensió teòrica, sinó també a les matemàtiques que la sustenten.

El codi desenvolupat es fa servir per comparar el rendiment de les diferents tècniques de compressió estudiades. Els resultats d'aquest estudi són coherents amb l'avenç de les tecnologies de compressió d'imatges en els darrers anys, amb el domini del mètode de compressió JPEG (Fourier) i posteriorment JPEG2000 (Wavelets).

Contents

1 Object **viii**

2 Abast **viii**

3 Requirements **viii**

4 Justification **ix**

5 Schedule **ix**

6 State of the art **1**

6.1 Lossles and lossy compression 1

6.2 JPEG 1

6.3 JPEG2000 2

7 Fourier transform **2**

7.1 Theoretical background 2

7.1.1 Analytic Fourier transform of sine and cosine waves 9

7.2 Discrete Fourier transform (DFT) 10

7.3 Fast Fourier transform (FFT) 10

7.3.1 Inverse Fast Fourier Transform (iFFT) 14

7.3.2 Matrix-Vector based FFT 15

7.3.3 The Cooley-Tukey Radix-2 Factorization 17

7.4 2D Fourier transform 17

7.5 Discrete cosine transform (DCT) 22

7.6 Image compression using Fourier transform 28

7.6.1 Filtering compression 28

7.6.2 Thresholding compression 29

7.6.3 JPEG compression 30

7.7 Short-Time Fourier Transform 32

7.8 Uncertainty Principle 35

8 Wavelet transform **36**

8.1 Theoretical background 36

8.2 Continuous Wavelet Transform 40

8.2.1 The CWT in terms of the FT 43

8.3 Discrete Wavelet Transform 45

8.3.1 Discretization of Scale and Translation Parameters 45
8.3.2 Multiresolution Analysis 46
8.3.3 Scaling Equation, Scaling Coefficients and Associated Wavelet Equation . 47
8.3.4 Haar Wavelet 48
8.3.5 Daubechies wavelets 50
8.3.6 Mallart Algorithm (wavelet packet decomposition) 51
8.4 Second-generation wavelet transform: Lifting scheme 51

9 Principal component analysis (PCA) 54
9.1 Singular value decomposition (SVD) 55
9.2 Image compression with PCA 58
9.3 Video compression with PCA 59
9.4 EigenFaces 61

10 Comparison between methods 63
10.1 Differences between images 63
10.2 Comparison between Fourier transform methods 65
10.2.1 Cut method for JPEG (threshold vs region) 69
10.3 Comparison between wavelet transform methods 69
10.4 Fourier, Wavelet and PCA comparison 72
10.5 Conclusions 75

A GIFs 76

List of Figures

1 Gantt Diagram for this project. (Source: own) ix
2 Example 1. Top left: Pure sine signal with 2 beats per second that last for 6 seconds. Top right: Fourier transform of a pure sine signal. Bottom: Signal x(t) wrapped in the complex plane for different winding frequencies. In a red box is the case at which the winding and the signal frequency match. 4
3 Example 2. Top left: Dyad signal made up by two pure sine waves ($x1$ & $x2$). Top right: Fourier transform of a dyad. Bottom: Dyad signal wrapped in the complex plane for different winding frequencies. In a red box the cases at which the winding and the signal frequency match. 6
4 Example 3. Top left: Composite function and the reconstruction (r) with 1% and 20% of its modes. Top right: Fourier transform of the composite function. Bottom: Composite function wrapped in the complex plane for different winding frequencies. 7

5 Example 4. Top left: Step function and the reconstruction (r) with 20% of its modes. Top right: Fourier transform of the step function. Bottom: Step function wrapped in the complex plane for different winding frequencies. 8

6 Butterfly diagram of a two element signal. Source own. 12

7 Radix-2 DIT FFT Cooleyt-Tukey algorithm (Butterfly diagram) for a length-8 signal. In this figure capital 'X' are in the frequency domain and non-capital 'x' are in time-space domain. Source: [6] 13

8 The single butterfly matrices B_i on the top; their products on the bottom. Source [22] 16

9 Mesh of an image of a cat. (If the GIF does not display, go to section A). Image source: [14]. Source own. 18

10 Schematic of 2D FFT. First, the FFT is taken of each row, and then the FFT is taken of each column of the resulting transformed matrix. Source: [5] 19

11 Vertical lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own. 20

12 Horizontal lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own. 20

13 Diagonal lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own. 20

14 Lines rotating clockwise and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own. 21

15 Moving dot and and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own. 21

16 Action of low and high pass filters on a image. Source: [39] 21

17 An image and its DCT coefficients. Source own 23

18 JPEG image division. JPEG transform compression starts by breaking the image into 8×8 groups, each containing 64 pixels. Three of these 8×8 groups are enlarged in this figure, showing the values of the individual pixels, a single byte value between 0 and 255. Source [34] 24

19 The 4 DCT Basis Functions of an 2-by-2 matrix. Source own. 25

20 The 14 DCT Basis Functions of an 4-by-4 matrix. Source own. 25

21 The 64 DCT Basis Functions of an 8-by-8 matrix. Source own. 26

22 The 256 DCT Basis Functions of an 16-by-16 matrix. Source own. 26

23 The DCT of the 14 DCT Basis Functions of an 4-by-4 matrix. Source own. . . . 27

24 The DCT of the 64 DCT Basis Functions of an 8-by-8 matrix. Source own. . . . 27

25 (a) Test pattern of size 688 × 688 pixels, and (b) its spectrum. The spectrum is double the image size as a result of padding, but is shown half size to fit. The circles have radii of 10, 30, 60, 160, and 460 pixels with respect to the full-size spectrum. Source: [19] 28

26 (a) Original image of size 688 × 688 pixels. (b)–(f) Results of filtering using ideal low pass filters with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Figure (25)(b). We used mirror padding to avoid the black borders characteristic of zero padding. Source [19] 28

27 (a) Original image of size 688 × 688 pixels. (b)–(f) Results of threshold compression removing the same number of information as each corresponding case of Figure (26). We used mirror padding to avoid the black borders characteristic of zero padding. Original image form [19]. Source own. 29

28 Example of run-length encoding. Each run of zeros is replaced by two characters in the compressed file: a zero to indicate that compression is occurring, followed by the number of zeros in the run. Source [34]. 30

29 JPEG quantization tables. These are two example quantization tables that might be used during compression. Each value in the DCT spectrum is divided by the corresponding value in the quantization table, and the result rounded to the nearest integer. Source [34]. 30

30 Serpentine pattern used in JPEG compression to convert the 8×8 DCT spectrum into a linear sequence of 64 values. Source [34]. 31

31 JPEG DCT-based encoder simplified diagram. Source [8]. 31

32 Illustration of short-time Fourier transform on the test signal $x(t)$ and a translating window $g(t)$. Source [18] 33

33 Illustration of the translating Gaussian window for the short-time Fourier transform. Source [5] 33

34 Audio signals. Source own. 34

35 Examples of spectograms of a sine wave and signals from Figure (34). Source own. 34

36 Illustration of resolution limitations and uncertainty in time-frequency analysis. Source: [5] 35

37 Dilation and translation of a wavelet. (a) Stretching and squeezing a wavelet: dilation ($a_1 = a_2/2; a_3 = 2 \cdot a_2$). (b)Moving a wavelet: translation. Source: [2] . . 37

38 The Fourier energy spectra of dilating wavelets. (a) Thre Mexican hat wavelets at three dilations, a = 0.5, 1.0, 2.0 and all located at b = 0. (b)Energy spectra corresponding to the wavelets in (a) all have identical energy and hence the same area under their curves. Note that only the positive part of the energy spectrum is shown. Source: [2] 37

39 Four wavelets. (a)Gaussian wave(first derivative of a Gaussian). (b)Mexican hat (second derivative of a Gaussian). (c)Haar. (d)Morlet (real part). Source: [2] . . 39

40 The fractal self-similiarity of the Daubechies mother wavelet. Source: [20] 39

41 Three Haar wavelets for the first two levels of the multi-resolution. Source: [5] . . 39

42 The wavelet of specific dilation and location on the signal. The regions which give positive and negative contributions to the integral are delineated in the sketch and marked with '+' and '-', respectively. Source: [2] 41

43 A wavelet interrogating a sinusoidal waveform: (a) The wavelet is in phase with the waveform giving good positive correlation. (b) The wavelet is now out of phase with the waveform giving good negative correlation. (c) The wavelet is again out of phase with the waveform, this time giving zero correlation. (d) The squeezed wavelet does not match the waveform locally. (e) The stretched wavelet does not match the waveform locally. Source: [2] 41

44 CWT plot of the signal above formed by two sines of different frequency concatenated. Source own. 42

45 Continious wavelet transform plot of two combined sinusoids. Top: A signal composed of a combination of two sinusoids with different frequencies. Bottom: Surface plot of $\mathcal{W}(a, b)$. Source own. 42

46 Continious wavelet transform plot of a single sinusoids. Left: Simple sinusoidal signal in time domain. Right: CWT plot of the signal . Source own. 43

47 Continious wavelet transform plot of a linear chirp signal. Left: Linear swept-frequency cosine signal. Right: CWT plot of the signal. Source own. 43

48 Schematic representation of the wavelet transform in its time and frequency representations. (a) The convolution of the wavelet with the signal. (b) The convolution in (a) expressed in the Fourier domain involves a product of the signal Fourier transform and the wavelet Fourier transform . Source: [2] 44

49 Discrete orthonormal wavelets: (a) The Haar scaling function in terms of shifted and dilated versions of itself. (b) The Haar wavelet in terms of shifted and dilated versions of its scaling function. (c) Three consecutive scales shown for the Haar wavelet family specified on a dyadic grid, for example, from top to bottom: $\psi_{m,n}(t)$,$\psi_{m+1,n}(t)$,$\psi_{m+2,n}(t)$. (d) Three Haar wavelets at three consecutive scales on a dyadic grid. (e) Three Haar wavelets at different scales. This time, the Haar wavelets are not defined on a dyadic grid and are hence not orthogonal to each other. Source: [2] 49

50 Procedure of a four-level signal decomposition using discrete wavelet transform. Note: H low-pass filter, G high-pass filter, A approximate information, D detailed information . Source: [2] 51

51 Forward transform procedure of lifting scheme. Source: [18] 53

52 Reverse transform procedure of lifting scheme. Source: [18] 53

53 An illustration of the forward transform. Step 1: the signal is split into even and odd parts. Step 2.1 and 2.2: the odd part is predicted linearly by the even part; the detail is what is missed by the prediction. Step 3: the calculated detail is used to update the even signal in order for the coarse signal to have the same average as the original signal. Source: [31] 53

54 An illustration of PCA where we project from 2D to 1D. Circles are the original data points, crosses are the reconstructions. The red star is the data mean. Code to generate the figure. Source: [28] 54

55 Illustration of the variance of the points projected onto different 1D vectors. v_1 is the first principal component, which maximizes the variance of the projection. v_2 is the second principal component which is direction orthogonal to v_1. Finally v' is some other vector in between v_1 and v_2. Code to generate the figure. Source: [28] 54

56 Principal components capture the variance of mean-subtracted Gaussian data shown in the left. The first three standard deviation ellipsoids (red), and the two left singular vectors, scaled by singular values shown in cyan. Image generate by code from: [5] 56

57 Distribution of points in the three main components. The red x's are the people with cancer and the blue circles are those who do not. Image generate by code from: [5] 57

58 Singular values and cumulative energy for the Ovarian cancer data. 57

59 Original image and the approximated with r elements. 58

60 Singular values and cumulative energy for the image from figure 59. 59

61 Singular values and cumulative energy for the video named *'VideoOriginal'*. . . . 60

62 The first 8 main components (eighenframes) for the video named *'VideoOriginal'* 60

63 Singular values and cumulative energy for the video named *'VideoOriginal'* for Case 1 and 2. 61

64 Some examples of the images used in this section. (This database can be downloaded here: Cat faces database) 61

65 The first 8 main components (eighenfaces) of the database used in this section . . 62

66 Singular values and cumulative energy for the database used in this section. . . . 62

67 Comparison of "Boat" images with different types of distortions, all with MSE = 210. (a) Original image (8bits/pixel; cropped from 512×512 to 256×256 for visibility); (b) Contrast stretched image, MSSIM = 0.9168; (c) Mean-shifted image, MSSIM = 0.9900; (d) JPEG compressed image, MSSIM = 0.6949; (e) Blurred image, MSSIM = 0.7052; (f) Salt-pepper impulsive noise contaminated image, MSSIM = 0.7748. Source: [38] 65

68 Original images used in this section with their sizes. Source image b): Stephen Hyde 66

69 Comparison of Fourier transform methods (FFT, DCT and JPEG) at different compression rates (15, 10, and 5 percent) for image a) of Figure (68). Source own. 67

70 Comparison of Fourier transform methods (FFT, DCT and JPEG) at different compression rates (15, 10, and 5 percent) for image b) of Figure (68). Source own. 68

71 The MSSIM for the three Fourier compression methods at different compression percentages for images a) and b) of Figure (68). Source own. 68

72 The MSSIM for the threshold and region cut methods at different compression percentages for image b) of Figure (68). Source own. 69

73 Comparison of wavelet transform methods (Matlab wavelet decomposition, DWT and lifting scheme) at different compression rates (25, 20, and 15 percent) for image a) of Figure (68). Source own. 70

74 The MSSIM for three wavelet compression methods at different compression percentages for images a) and b) of Figure (68). Source own. 71

75 Comparison of wavelet transform methods (Matlab wavelet decomposition, DWT and lifting scheme) at different compression rates (25, 20, and 15 percent) for image b) of Figure (68). Source own. 71

76 The MSSIM for the two best performing wavelet compression methods at different compression percentages for images a) and b) of Figure (68). Source own. 72

77 Comparison between image compression techniques (JPEG, multiresolution lifting scheme and PCA) at different compression rates (15, 10, and 5 percent) for image a) of Figure (68). Source own. 73

78 The MSSIM for the three methods (JPEG, multiresolution wavelet decomposition and PCA) at different compression percentages for images a) and b) of Figure (68). Source own. 73

79 Comparison between image compression techniques (JPEG, multiresolution lifting scheme and PCA) at different compression rates (15, 10, and 5 percent) for image b) of Figure (68). Source own. 74

80 Comparison between JPEG and multiresolution wavelet packet decomposition at similar MSZSIM scores for image a) of Figure (68) 74

List of Tables

1 Tasks to be performed in order to achieve the objective of the project. x

2 Standard deviation of data and normalized singular values. 56

1 Object

The goal of this project is to develop an in-house code that performs image and signal compression using machine learning techniques (Fourier, wavelets, and PCA). The project's code intelligibility will become increasingly important as the complexity of the project increases. Therefore, object-oriented programming will play a significant role in this study.

2 Abast

The following concepts will be studied and implemented in an in-house code as part of the project's development:

- Fourier Transform and Fast Fourier Transform
- Wavelet Transform
- Principal Component Analysis (PCA)
- Comparative in compression performance of the different methods studied

All these will be enclosed and performed in:

- MATLAB language and environment
- Object-Oriented Programming (OOP) paradigm

This project will not include:

- Development of SVD libreries
- Development of matrix-vector multiplication

High level deliverables:

- Project report
- Budget
- Code files

3 Requirements

- The TFG dedication limit is 600 hours.
- All academic documents will be written in English.
- The project's code must be developed in MATLAB.
- The code will be developed using object-oriented programming
- GitHub will be used for version control and source code management (SCM).
- The academic deadline for all the deliverables is June 22nd.
- The results will be validated with other certified libreries (FFTW, Wavelets, etc.)

4 Justification

In the age of the Internet and cloud-based applications, image compression has become increasingly important. According to Rise Above Research, a market research firm for the digital imaging industry, the global population will have taken 1.4 trillion photos. And by 2023, this figure is expected to reach 1.61 trillion. In 2021, 7.88 trillion photos are expected to be stored, rising to 9.94 trillion by 2023. [29]

The stored and processed images must be compressed and recovered with minimal loss. Image compression theory is becoming increasingly important to reduce data redundancy and save more space on hardware and transmission bandwidth.Natural images have a high amount of redundant information; neighboring pixels tend to correlate with each other [24]. Most image compression algorithms take advantage of this correlation to reduce the size of stored images.

Fast and efficient compression methods are necessary not only for images we take with our cell phones, but also in the fields like of telemedicine, satellite data, machine learning, and other similar applications. Image compression can be done using machine learning and deep learning. In this project, only machine learning is developed. A study into the sparsity of information after compression and the efficiency of reconstruction will be performed to compare different techniques.

5 Schedule

The project was started approximately mid-September to better distribute the workload.

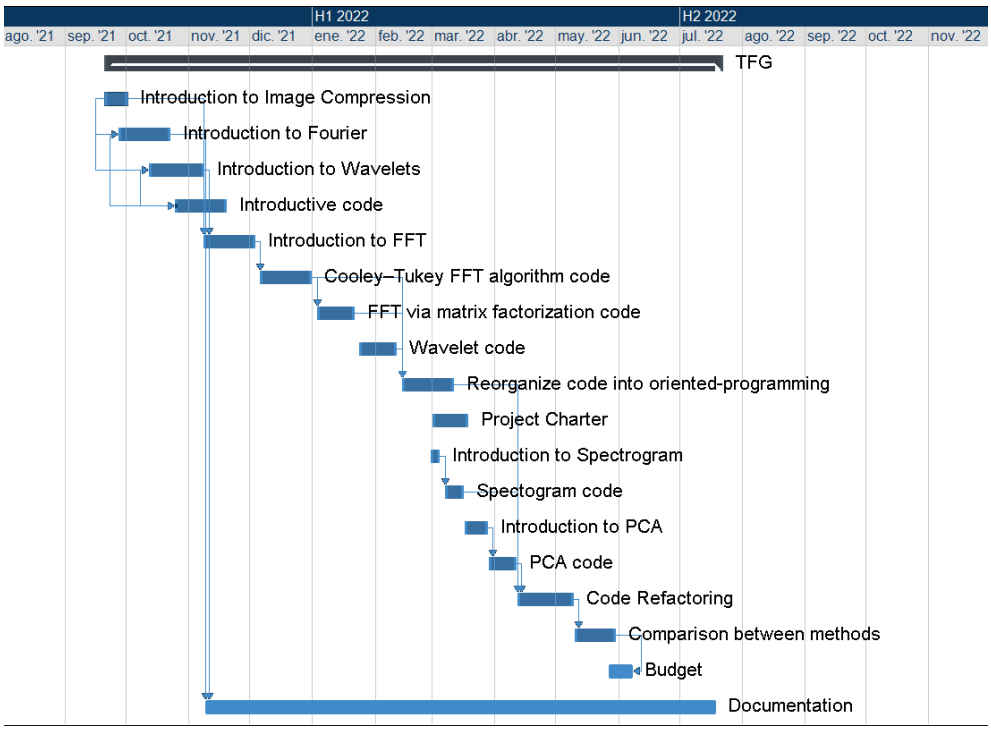


Figure 1 Gantt Diagram for this project. (Source: own)

Table 1 Tasks to be performed in order to achieve the objective of the project.

Nº	**Task nam**	**Description**
1.	Fourier	
1.1	Fourier Transform	Study and develop a code that performs Fourier Transforms
1.2	Fast Fourier Transform	Study and develop a code that performs FFT
1.3	Fast Fourier Transform via matrix factorization	Study and develop a code that performs FFT via matrix factorization
1.4	Short Time Fourier Transform	Study and develop a code that performs STFT
2.	Wavelets	
2.1	Wavelets transform	Study and develop a code that performs Wavelets transforms
3.	PCA	Study and develop a code that performs PCA
4.	Code development	Develop and Object-Oriented Programming (OOP) code
5.	Project managment	Report, budget, project charter, and deliverables

6 State of the art

The goal of image compression is to reduce the irrelevance and redundancy of image data in order to store or transmit data efficiently. The focus is on reducing the number of bits required to represent an image. There are two types of image compression: lossy and lossless.

6.1 Lossles and lossy compression

Lossless compression permits perfect recuperation of original information with no data loss. No single lossless compression technique is capable of compressing all types of potential data. Depending on the type of information we want to compress, there are several methods that are designed to expect patterns in the input data or in the type of redundancy that may be present. Lossless compression methods are reversible. Medical imaging, technical drawings, clip art, and comics are common applications for lossless compression. Examples of file formats that support lossless compression include GZIP, Brotli, WebP, and PNG.

Lossy compression, on the other hand, uses imprecise approximations and partial data removal to represent the information; therefore, this type of data compression is not reversible. In exchange for this distortion in reconstruction, we can often achieve much higher compression ratios than lossless compression. Lossy algorithms are particularly advantageous for natural images such as photographs, where the small (often undetectable) loss of authenticity is acceptable to achieve a significant reduction in bit rate. The discrete cosine transform (DCT) and the wavelet transform are the most commonly used lossy compression methods and are used by the JPEG and JPEG2000 file formats, respectively. In this project, only lossy compression will be studied.

6.2 JPEG

JPEG is the most commonly used method of lossy compression for digital images. This method was introduced in 1992 by the Joint Photographic Experts Group (source of the acronym) with the intention to establish a standard. JPEG is based on the discrete cosine transform (DCT) introduced by Nasir Ahmed in 1972. JPEG is largely responsible for the widespread use of digital photos on the Internet and social media, as it allows faster and cheaper transmission of images around the world. The amount of compression that JPEG performs can be customized, allowing for an adjustable trade-off between image quality and storage space. JPEG typically achieves a compression ratio of 10:1, with little perceptible degradation of image quality. Depending on the image content, acceptable results can be achieved at compression ratios of up to 120:1 [21]. The JPEG method was proposed in *CCITT Recommendation T.81: Terminal Equipment and Protocols for Telematic Services: Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines* and can be found in the bibliography of this thesis.

6.3 JPEG2000

JPEG200 is the standard compression method successor to JPEG, developed from 1997 to 2000. This compression method uses the wavelet transform which exhibits better results. This method introduces multilevel analysis which can overcome the difficulties of the JPEG method. The JPEG200 was chosen in 2004 by the Digital Cinema Initiatives (DCI) as the new standard for digital cinema compression [35]. Although JPEG2000 offers improvements in compression performance, some drawbacks hinder its implementation. Currently, more than 20 years after its introduction, JPEG2000 is only supported in the Safari web browser. Two of the most important drawbacks of JPEG2000 are the following. First of all, the encoding procedure of JPEG2000 is slower and more complex than JPEG. Secondly, too high image compression can greatly distort an image making it unusable. This compression limit depends on each image so sometimes the only way to achieve high compression is to manually set this limit. This effect does not happen in JPEG.

7 Fourier transform

The Fourier transform is a very powerful mathematical tool that allows us to visualize information from another point of view where complex problems can have a simple solution. The basic idea behind compression through transforms is that after the Fourier transform, or other, not all information is of equal importance. [34]

The French mathematician Joseph Fourier, in his book *The Analytical Theory of Heat* published in 1822, asserted that any signal can be represented by a weighted integral of a series of sine and cosine functions. The Fourier transform is the name for such an integral, and it is probably the most widely used signal processing tool in science and engineering. [15]

7.1 Theoretical background

The Fourier transform is a mathematical method that decomposes functions depending on space or time into functions depending on spatial or temporal frequency. It is reversible, being able to transform from time to frequency and from frequency to time. Using the notation of inner product, the Fourier transform can be expressed as

$$\hat{x}(f) = \langle x \, , e^{i2\pi ft} \rangle = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt, \tag{1}$$

and the inverse Fourier transform is defined as

$$x(t) = F^{-1}[\hat{x}(f)] = \langle \hat{x}(f) \, , e^{-i2\pi ft} \rangle = \int_{-\infty}^{\infty} \hat{x}(f) e^{i2\pi ft} df. \tag{2}$$

Example 1: Simple signal

To understand how this mathematical tool works, first of all, we will start with a pure simple signal and a finite time interval. For example, a signal $x(t)$ with 2 beats per second that lasts for 6 seconds. See Figure (2).

As can be seen in the definition of the Fourier transform, the signal $x(t)$ is multiplied with complex exponential $e^{-2\pi ift}$, which is the key point. The idea is to wrap the function $x(t)$ around a circle in the complex plane with a winding frequency f. To better visualize it, imagine a rotating position vector where its length is equal to the height of the signal for each point in time. High points of the graph correspond to a greater distance from the origin of the complex plane, and points close to zero end up closer to the origin. Low points correspond to a negative distance from the origin. Now the frequency with which we're wrapping the graph around the circle can be changed by changing the winding frequency f. [30]

Keep in mind that there are two different frequencies, the frequency of the signal and the winding frequency. When these two frequencies match, all the high points of the graph coincide at the same point in the complex plane, and similarly for all the low points. If the center of mass ($c.m.$) is calculated for the wrapped graph, for almost all winding frequencies, the center of mass stays close to the origin, but when the frequency of the signal and the winding frequency match, the center of mass is exceptionally far from the origin of the complex plane.

The simple way to calculate this center of mass is to just calculate the average position on the complex plane of a finite number of points. To obtain a more precise calculation, just more points have to be sampled. In the limit, the summation transforms into an integral of the function $x(t) \cdot e^{-2\pi ft}$ divided by the size of the time interval that is been looked at, this is

$$c.m. = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} x(t) e^{-2\pi ift} dt. \tag{3}$$

Using Euler's formula

$$e^{-ix} = \cos(x) - i \sin(x), \tag{4}$$

a relationship can be established between the complex exponential function and the trigonometric functions. Thus, the c.m. is defined as

$$c.m. = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} x(t) \left[\cos(2\pi ft) - i \sin(2\pi ft)\right] dt.$$

The Fourier transform does not divide the integral by the size of the time integral, instead, it is obtaining the position of the center of mass scaled up by a factor of time. Essentially, this has the effect of scaling up the magnitude of the Fourier transform at a given frequency as it persists for an extended period of time.

In the definition of the Fourier transforms, the boundary of the integral is set to $-\infty$ and ∞, although in practice, with things like sound or image processing, we integrate over a finite time interval and use the discrete Fourier transform, which can be expressed as matrix-vector multiplication. This type of Fourier transform will be studied in chapter 7.2.

In this link, you can access a tool where you can visualize an interactive animation of a Fourier transform.

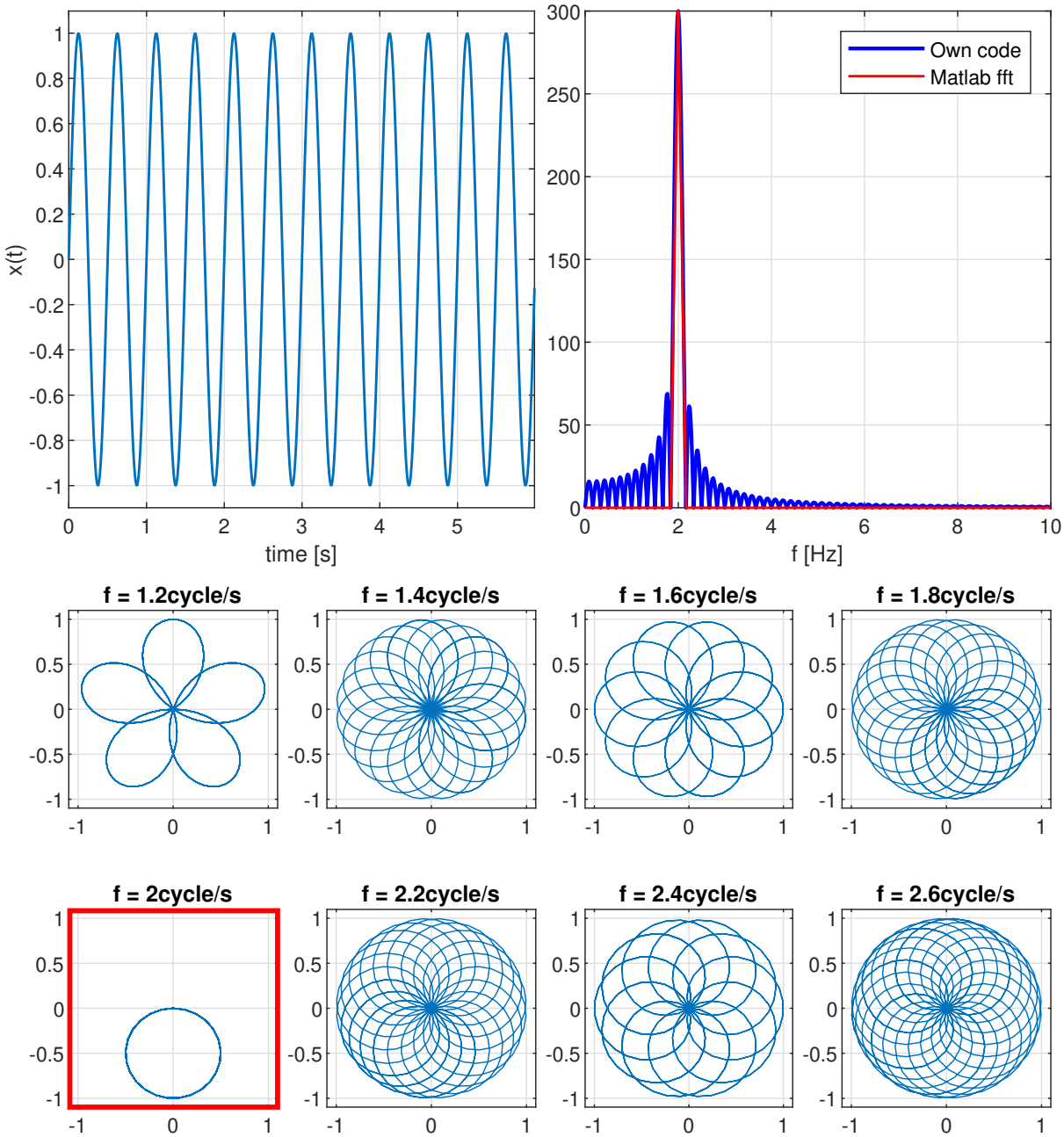


Figure 2 Example 1. Top left: Pure sine signal with 2 beats per second that last for 6 seconds. Top right: Fourier transform of a pure sine signal. Bottom: Signal x(t) wrapped in the complex plane for different winding frequencies. In a red box is the case at which the winding and the signal frequency match.

Example 2: Composition of sines

The Fourier transform can be also used for simple signals but also for signals composed of other signals. Our task is to decompose a signal $x(t)$ into the pure frequencies that make it up. For example, the recording of a chord of pure notes, where the microphone records the air pressure at many different points in time and we want to know the frequencies (notes) that make up the chord. In this case, we'll use a two-note chord known as a dyad. The winding signal in the complex plane will have the center of mass exceptionally far from the origin for each of the matching frequencies of the signals that it is composed of. See Figure (3).

Example 3: Composite signal

More generally, the Fourier transform can transform any signal into the frequency domain. In Figure (4), it can be seen how a signal that is not composed of a sum of sine and cosine waves can be transformed and its main frequency can be determined. Sometimes some oscillations appear in the reconstructed signal, typically around a jump discontinuity. It is known as the Gibbs phenomenon and will be studied in more detail in the next example.

Example 4: Step signal

The purpose of this example is a brief introduction to the Gibbs phenomenon. To do so, a step function is going to be considered. In Figure (5), we see that the reconstruction of the signal produces large peaks around the jump discontinuity which overshoot and undershoot the function's actual values. The Gibbs phenomenon exhibits the challenging problem of approximating a discontinuous function with a finite series of continuous sinusoidal waves. The peaks around the discontinuity become narrower as the number of sines for signal reconstruction increases. It can be proven that if periodic function $x_p(t)$ with a jump discontinuity of J_K, as the number of sines for signal reconstruction increases to infinity, the Fourier reconstruction of the function returns not only the function but also a pair of straight lines that extend equally above and below the discontinuity by an amount Ak specified by: [4]

$$A_k = \left[\frac{1}{2} - \frac{1}{\pi}\int_0^{\pi} \frac{sen(\xi)}{\xi}\, d\xi\right] J_k = 0.0895 J_k. \tag{5}$$

Therefore, the lines extend about 9 percent. This finding is consistent with the case of the figure 5 where the magnitude of the step is one and the magnitude of the oscillations have an amplitude A_k lower than 0.1 but extremely near to it.

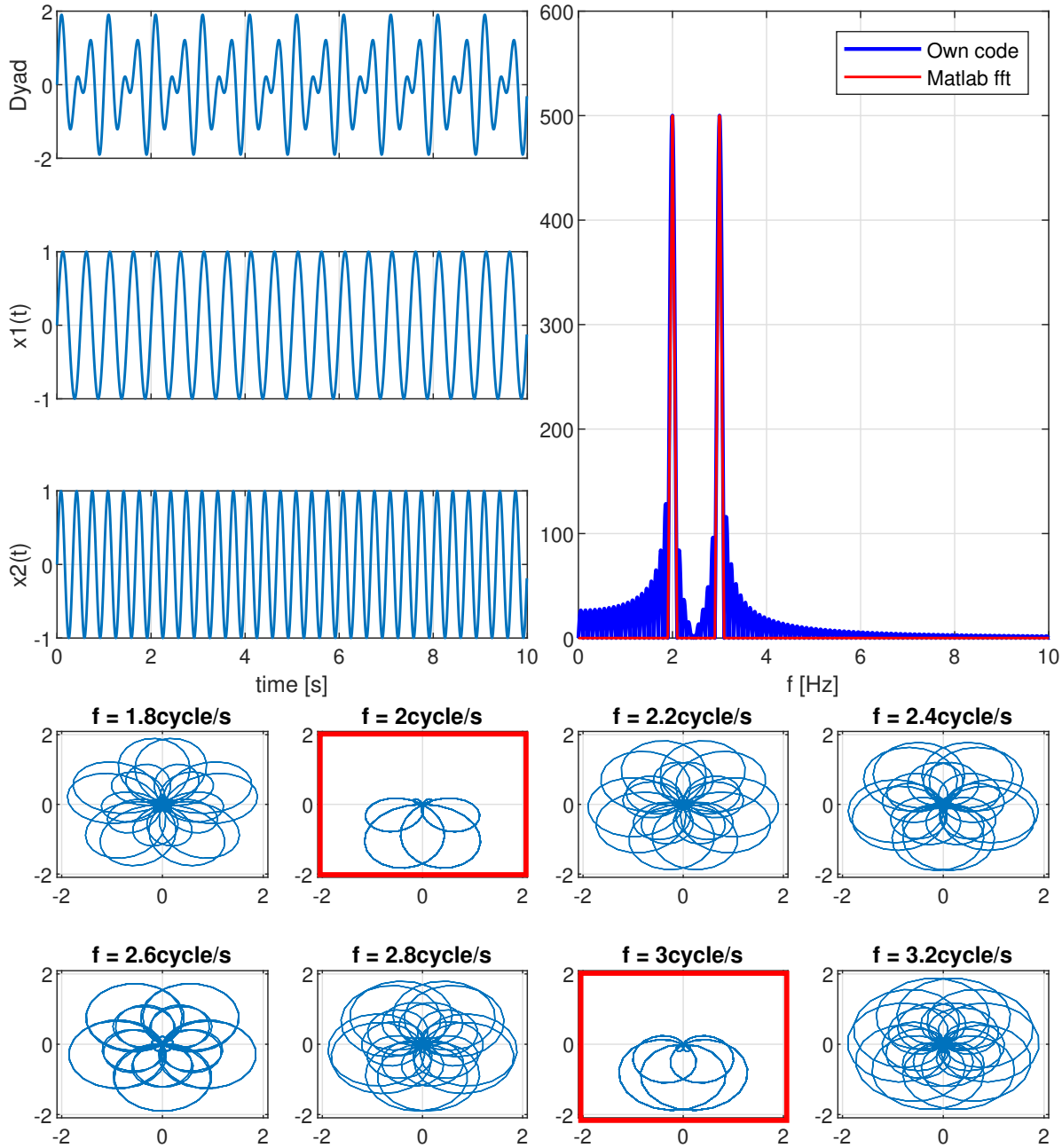


Figure 3 Example 2. Top left: Dyad signal made up by two pure sine waves ($x1$ & $x2$). Top right: Fourier transform of a dyad. Bottom: Dyad signal wrapped in the complex plane for different winding frequencies. In a red box the cases at which the winding and the signal frequency match.

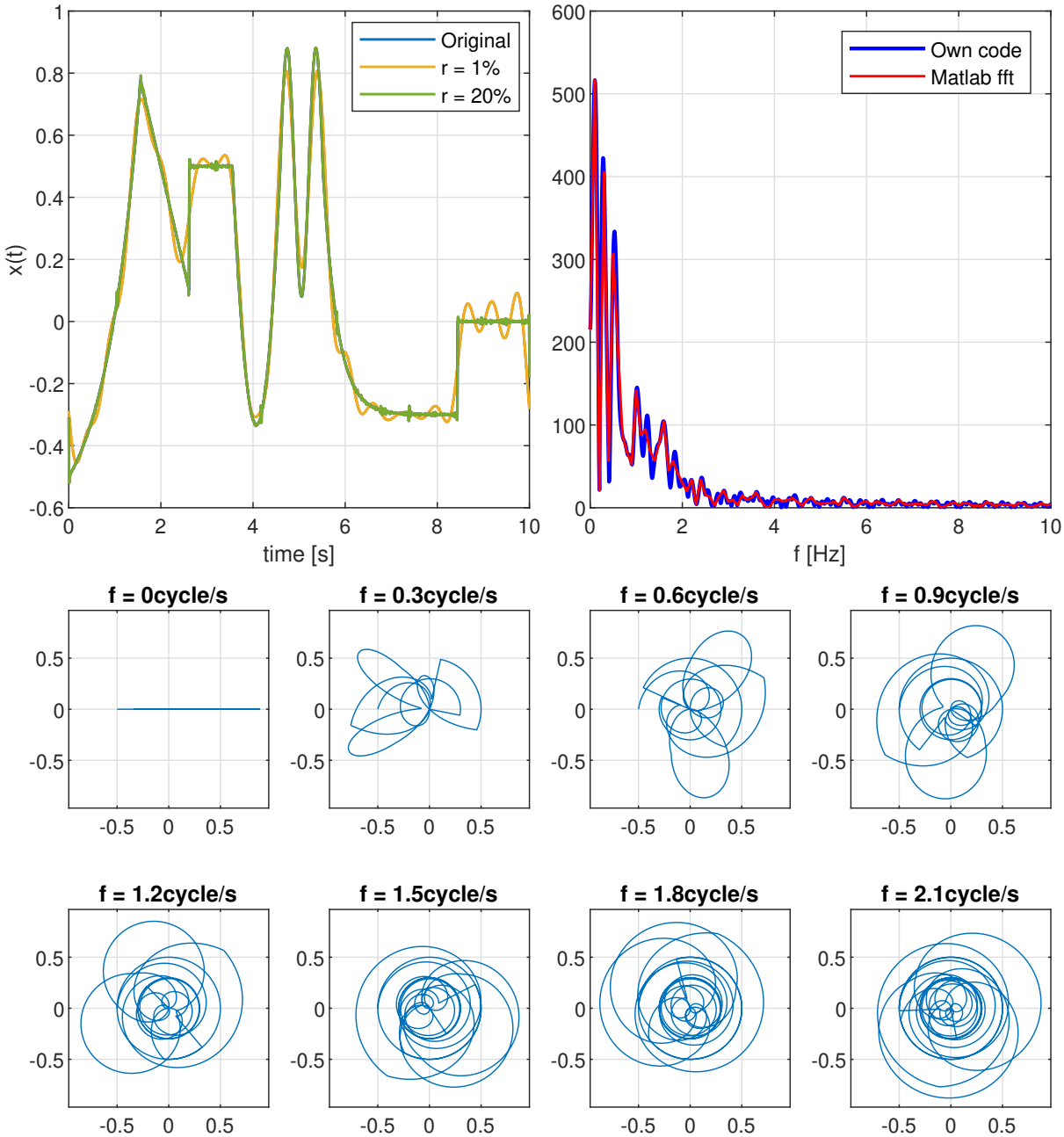


Figure 4 Example 3. Top left: Composite function and the reconstruction (r) with 1% and 20% of its modes. Top right: Fourier transform of the composite function. Bottom: Composite function wrapped in the complex plane for different winding frequencies.

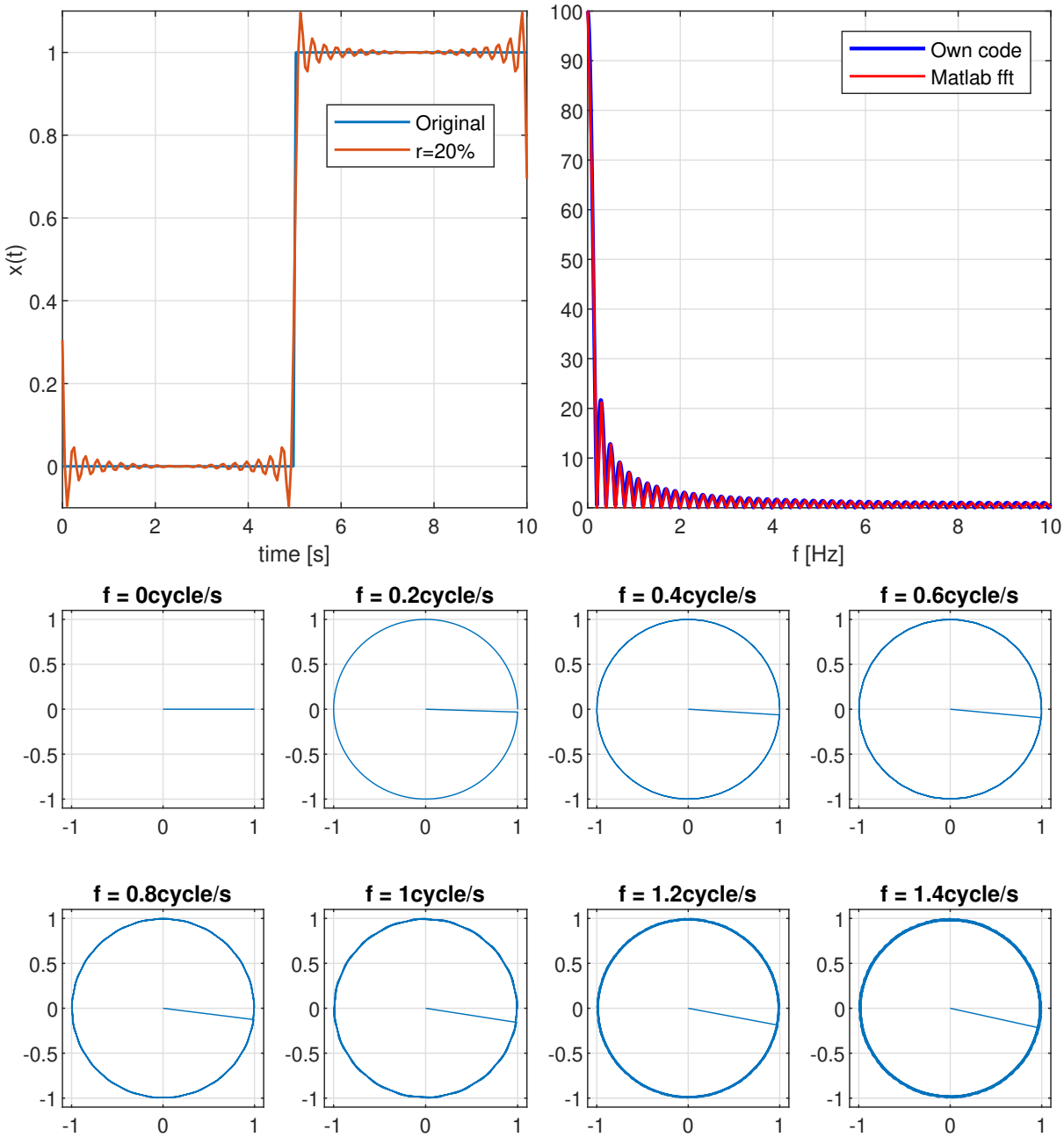


Figure 5 Example 4. Top left: Step function and the reconstruction (r) with 20% of its modes. Top right: Fourier transform of the step function. Bottom: Step function wrapped in the complex plane for different winding frequencies.

7.1.1 Analytic Fourier transform of sine and cosine waves

Fourier transforms for the sine and cosine function can be easily obtained using the Fourier Transform of the complex exponential. To do so, the sine and cosine functions can be rewritten using Euler's identity (4) as

$$\cos(2\pi at) = \frac{e^{i2\pi at} + e^{-i2\pi at}}{2}, \tag{6}$$

$$\sin(2\pi at) = \frac{e^{i2\pi at} - e^{-i2\pi at}}{2i}. \tag{7}$$

To calculate the Fourier transform of these equations, we first need to calculate the Fourier transforms of the complex exponential. To do so we are going to start from the Fourier transform of a Dirac-Delta Impulse function.

$$\mathcal{F}\{\delta(t-a)\} = \int_{-\infty}^{\infty} \delta(t-a)e^{-i2\pi ft}\,dt = e^{-i2\pi fa}.$$

This result is obtained using the shifting property of the Dirac-Delta:

$$\int_{-\infty}^{\infty} \delta(t-a)g(t)\,dt = g(a).$$

The inverse Fourier transform of the complex exponential is the impulse,

$$\mathcal{F}^{-1}\{e^{-i2\pi fa}\} = \int_{-\infty}^{\infty} e^{-i2\pi fa}e^{i2\pi ft}\,df = \int_{-\infty}^{\infty} e^{i2\pi f(t-a)} = \delta(t-a).$$

Similarly, the Fourier transform of a complex exponential is

$$\mathcal{F}\{e^{i2\pi at}\} = \int_{-\infty}^{\infty} e^{i2\pi at}e^{-i2\pi ft}\,dt = \delta(f-a).$$

Finally, the Fourier transforms of sine and cosine functions using (6) and (7) can be obtained as

$$\mathcal{F}\{\cos(2\pi at)\} = \int_{-\infty}^{\infty} \frac{e^{i2\pi at} + e^{-i2\pi at}}{2} e^{-i2\pi ft}\,dt$$

$$= \frac{1}{2}\left[\int_{-\infty}^{\infty} e^{i2\pi at}e^{-i2\pi ft}\,dt + \int_{-\infty}^{\infty} e^{-i2\pi at}e^{-i2\pi ft}\,dt\right] = \frac{1}{2}\left[\delta(f-a) + \delta(f+a)\right],$$

$$\mathcal{F}\{\sin(2\pi at)\} = \int_{-\infty}^{\infty} \frac{e^{i2\pi at} - e^{-i2\pi at}}{2i} e^{-i2\pi ft}\,dt$$

$$= \frac{1}{2i}\left[\int_{-\infty}^{\infty} e^{i2\pi at}e^{-i2\pi ft}\,dt - \int_{-\infty}^{\infty} e^{-i2\pi at}e^{-i2\pi ft}\,dt\right] = \frac{1}{2i}\left[\delta(f-a) - \delta(f+a)\right].$$

7.2 Discrete Fourier transform (DFT)

Approximating the Fourier transform on discrete vectors of data is required when computing or working with real-data. Although we will always use the Fast Fourier transform (FFT) for computations we will start from the the Discrete Fourier transform (DFT).

The DFT is basically a discretized version of the Fourier transform for vectors of data $x = [x_1, x_2, x_3, \cdots, x_n]^T$ obtained by discretizing the function $x(t)$ at regular intervals Δt.

The DFT is given by

$$\hat{x}[k] = \sum_{j=0}^{n-1} x(j) e^{-\frac{i2\pi jk}{n}} \qquad k = 0, 1, ..., n-1 \tag{8}$$

and the inverse Discrete Fourier transform (iDFT) is given by

$$x[k] = \frac{1}{N} \sum_{j=0}^{n-1} \hat{x}(j) e^{\frac{i2\pi jk}{n}} \qquad k = 0, 1, ..., n-1. \tag{9}$$

The DFT is a linear operator that represents the data using sine and cosine functions with integer multiples of a fundamental frequency, $\omega_n = e^{-2\pi i/n}$. The DFT may be computed by matrix multiplication:

$$\hat{x} = Fx \tag{10}$$

$$\begin{bmatrix} \hat{x}[1] \\ \hat{x}[2] \\ \hat{x}[3] \\ \vdots \\ \hat{x}[n] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} x[1] \\ x[2] \\ x[3] \\ \vdots \\ x[n] \end{bmatrix} \tag{11}$$

7.3 Fast Fourier transform (FFT)

The DFT is extremely useful for computation and numerical approximation, but it does not scale well to very large n because it implies a multiplication of a dense $n \times n$ matrix, resulting in a method $O(N^2)$. Using the FFT we can reduce the problem to $O(Nlog_2N)$.

In Matlab, the FFT functions (fft, fft2, fftn, ifft, ifft2, ifftn) are based on a library called FFTW. This library uses the well-known Cooley-Tukey algorithm [16, 11]. This algorithm is based on the following observation:

$$\omega_{2n}^2 = \omega_n \tag{12}$$

where ω is the kernel of the Fourier transform:

$$\omega_n^{jk} = e^{\frac{i2\pi jk}{n}}$$

where j and k are integers that swipe through different frequency and spatial components. Taking $j = k = 1$ it's easy to see the property mentioned before.

$$\omega_{2n}^2 = e^{\frac{i2\pi}{2n}} \cdot e^{\frac{i2\pi}{2n}} = e^{\frac{i2\pi}{n}} = \omega_n. \tag{13}$$

We can use this property to simplify the Fourier transform because the matrix F depends only on ω, n, j, k. The basic idea is to separate the $x[n]$ into even and odd-indexed subsequences.

$$\hat{x}[k] = \sum_{n=0}^{N-1} x[n]\omega_N^{kn} = \sum_{n_{even}} x[n]\omega_N^{kr} + \sum_{n_{odd}} x[n]\omega_N^{kr}$$

$$n = \begin{cases} 2r & \text{even} \\ 2r+1 & \text{odd} \end{cases}$$

$$r = 0, 1, ..., \frac{N}{2} - 1,$$

$$\hat{x}[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\omega_N^{k2r} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\omega_N^{k(2r+1)} = \sum_{r=0}^{\frac{N}{2}-1} x[2r](\omega_N^2)^{kr} + \omega_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1](\omega_N^2)^{kr}. \tag{14}$$

Using the property of equation (13), equation (14) can be expressed as:

$$\omega_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j\frac{2\pi}{N/2}} = \omega_{N/2}$$

$$\hat{x}[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\omega_{N/2}^{kr} + \omega_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\omega_{N/2}^{kr}. \tag{15}$$

The first summation is an N/2 point DFT associated with the even samples of $\hat{x}$. We are going to call these coefficients $\hat{x}_e[k]$. The second summation is also an N/2 point DFT associated with the odd sample of $\hat{x}$. Analogously we are going to call these coefficients $\hat{x}_o[k]$. Now the DFT can be written as

$$\hat{x}[k] = \hat{x}_e[k] + \omega_N^k \hat{x}_o[k]. \tag{16}$$

Thanks to the periodicity of the complex exponential, $\hat{x}[k + \frac{N}{2}]$ can also be obtained from $\hat{x}_e[k]$ and $\hat{x}_o[k]$

$$\hat{x}[k + \frac{N}{2}] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\omega_{N/2}^{r(k+\frac{N}{2})} + \omega_N^{k+\frac{N}{2}} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\omega_{N/2}^{r(k+\frac{N}{2})}$$

$$\hat{x}[k + \frac{N}{2}] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]e^{-\frac{2\pi i}{N/2}r(k+\frac{N}{2})} + e^{-\frac{2\pi i}{N}(k+\frac{N}{2})} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]e^{-\frac{2\pi i}{N/2}r(k+\frac{N}{2})}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r]e^{-\frac{2\pi i}{N/2}rk}e^{-2\pi ri} + e^{-\frac{2\pi i}{N}k}e^{-\pi i}\sum_{r=0}^{\frac{N}{2}-1} x[2r+1]e^{-\frac{2\pi i}{N/2}rk}e^{-2\pi ri}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r]e^{-\frac{2\pi i}{N/2}rk} - e^{-\frac{2\pi i}{N}k}\sum_{r=0}^{\frac{N}{2}-1} x[2r+1]e^{-\frac{2\pi i}{N/2}rk}$$

$$\hat{x}[k+\frac{N}{2}] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\omega_{N/2}^{kr} - \omega_N^k\sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\omega_{N/2}^{kr}$$

$$\hat{x}[k+\frac{N}{2}] = \hat{x}_e[k] - \omega_N^k\hat{x}_o[k]. \tag{17}$$

The order of this problem is $O(N^2/2+N)$, so we've roughly cut the calculations by a factor of 2 using this approach. But why stop here, now we can split this problem in half, and if N is a power of 2 we can keep splitting recursively until we have a DFT of only 1 point. To reach this level we need to split the problem log_2N times. Now the number of calculations needed are $N + Nlog_2N$ and for large numbers, we can say that this problem is order $O(Nlog_2N)$.

The base of this algorithm is the butterfly diagram applied to each of the matrices obtained. The butterfly diagrams show the data flow for each element of the matrix before, during, and after each FFT. As seen before, given a two-point vector $x[0]$ and $x[1]$, we can find the FFT with the following equations:

$$\hat{X}[0] = x[0] + \omega_2^0 x[1]$$
$$\hat{X}[1] = x[0] - \omega_2^0 x[1].$$

This system of equations can be represented with the butterfly diagram shown in Figure (6), named for its cross structure.

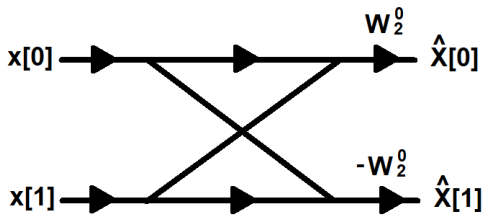


Figure 6 Butterfly diagram of a two element signal. Source own.

To calculate the FFT of more elements we simply have to apply this simple butterfly diagram sequentially by levels as seen in Figure (7). As can be seen, the input is bit-reversed while the output is in natural order. This is called decimation-in-time (DIT). There are decimation-in-frequency (DIF) algorithms in which the input is in natural order while the output is bit reversed order. Both algorithms require $Nlog_2N$ operations.

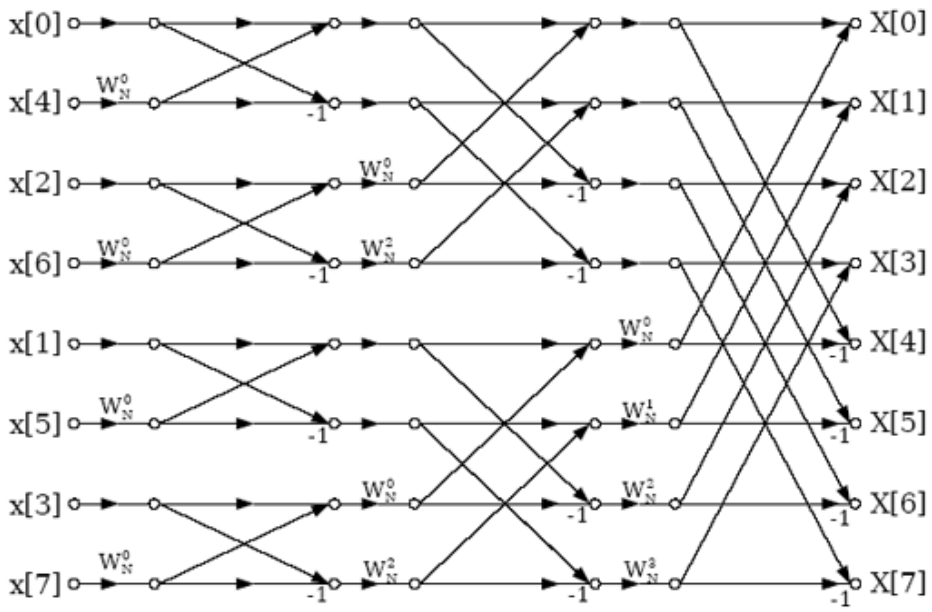


Figure 7 Radix-2 DIT FFT Cooleyt-Tukey algorithm (Butterfly diagram) for a length-8 signal. In this figure capital 'X' are in the frequency domain and non-capital 'x' are in time-space domain. Source: [6]

In order to perform the FFT using the Cooley-Tukey algorithm, the data must be a power of two to be able to divide between even and odd recursively. It is a radix-2 algorithm[1]. If the input signal is not a power of 2 there are several ways to perform the FFT. The simplest solution is to fill the signal with zeros to the nearest power of two known as zero-padding. This also increases the resolution of the study signal allowing you to find frequencies that may have been hidden before. This practice can also lead to problems due to the continuity jump at the ends of the signal. This type of padding leads to black bordes in the reconstruction images. Another type of padding is using a mirror signal. More complex but more direct solution to this problem is to implement radix-3 and radix-5 algorithms and use them in combination with radix-2 in order to solve vectors of different lengths.

There are other algorithms that are also capable of performing FFTs, eg. prime-factor algorithm (PFA), Bruun's algorithm, and Rader's algorithm. There are also some specialized algorithms for real data only. Although they could be useful for image compression, these algorithms will not be studied in this thesis.

[1]Radix describes the number of digits utilized in a positional number system before "rolling over" to the next digit's place.

7.3.1 Inverse Fast Fourier Transform (iFFT)

There are different methods to calculate the iFFT but for this project but we are going to use the method described in *Verified Signal Processing Algorithms in Matlab and C* by Arie Dickman where the iFFT can be computed from any Fourier transform method as [12]

$$IFFT(X) = \frac{1}{N} conj(FFT(conj(X))). \tag{18}$$

The derivation of this method will be shown below [32].

The first step is to realize that the DFT and iDFT are almost the same (see equations (8) and (9)). The only differences between them are the scaling factor of $1/N$ and the sign of the exponential. Therefore, using the complex conjugate, the exponential term of the DFT can be rewritten:

$$\hat{x}[k] = \sum_{j=0}^{n-1} x(j) conj\left(e^{-\frac{i2\pi jk}{n}}\right) \qquad k = 0, 1, ..., n-1. \tag{19}$$

Using the following properties of complex conjugates, the complex conjugate operator can be pulled outside of the sum, this is

$$conj(a)conj(b) = conj(ab), \tag{20}$$

$$conj(a) + conj(b) = conj(a + b). \tag{21}$$

To ease the intelligibility of the equations and the work with them, the sign '$\hat{x}$' will be changed to '$DFT(x)$'. Hence, equation (19) can be transformed into:

$$DFT(conj(x[k])) = \sum_{j=0}^{n-1} conj\left(x(j)e^{-\frac{i2\pi jk}{n}}\right) \qquad k = 0, 1, ..., n-1, \tag{22}$$

$$DFT(conj(x[k])) = conj\left(\sum_{j=0}^{n-1} x(j)e^{-\frac{i2\pi jk}{n}}\right) \qquad k = 0, 1, ..., n-1. \tag{23}$$

The right side of the equation matches the definition of the iDFT by a factor of N so it can be substituted by

$$DFT(conj(x[k])) = conj\left(N \cdot iDFT(x)\right). \tag{24}$$

To get to the final result with simple algebra we obtain

$$DFT(conj(x[k])) = Nconj\left(iDFT(x)\right),$$

$$\frac{1}{N} DFT(conj(x[k])) = conj\left(iDFT(x)\right),$$

$$\frac{1}{N} conj(DFT(conj(x[k]))) = iDFT(x). \tag{25}$$

The outputs obtained by the FFT and iFFT are the same exact results like the ones obtained by DFT and iDFT. The only thing that changes is that the technique used to compute the results requires less computation time. Therefore it is possible to exchange the DFT for an FFT thus arriving at equation (18).

7.3.2 Matrix-Vector based FFT

As seen in section 7.3 in the Cooley-Tukey algorithm, there is a relationship between ω_{2n}^2 and ω_n that allows us to divide the problem into small and easy to solve problems. However, for faster and more efficient implementation in computers, the problem can be solved with a Matrix-Vector approach. [22, 25]

The basic idea of this section is to associate the F_N matrix to the smaller $F_{N/2}$ matrix. Therefore, an n-point DFT can be calculated from two faster and more efficient (n/2)-point DFTs. The radix-2 FFT is based on the idea of repeating this process. For the DIT-FFT, the following equation is shown:

$$F_N^{'} = F_N P_N^T = \begin{bmatrix} F_{N/2} & \Omega_{N/2} F_{N/2} \\ F_{N/2} & -\Omega_{N/2} F_{N/2} \end{bmatrix} \tag{26}$$

$$\Omega_N/2 = diag(\omega_N^{(n-1)}), \qquad n = 1, ..., N/2$$

where P_N is a $N \times N$ permutation matrix that performs an even-odd ordering of the rows when multiplied from the left by P_N or columns if it is multiplied from the right by P_N^T. For easier understanding and application of this relationship, the matrix from equation (26) can be decomposed into the product of two matrices.

$$F_N^{'} = \begin{bmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{bmatrix} \cdot \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \tag{27}$$

where the $I_{N/2}$ matrices are identity matrices of size $N/2$. The FFT butterfly operations described in section 7.3 are represented in the first matrix, whereas two separate DFTs of length N/2 are included in the second matrix generating a block diagonal matrix.

Equation (27) can be used to calculate the Fourier transform of a vector x_n. First of all, it will be necessary to perform an even-odd rearrange of x_n.

$$P_N \cdot x_N = \begin{bmatrix} x_{odd} \\ x_{even} \end{bmatrix} \tag{28}$$

Thanks to the fact that $P_N^T \cdot P_N = I_N$, is it possible to transform equation (10) into:

$$\hat{x}_N = F_N \cdot x_N = F_N \cdot P_N^T \cdot P_N \cdot x_N = \begin{bmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{bmatrix} \cdot \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \cdot \begin{bmatrix} x_{odd} \\ x_{even} \end{bmatrix} \tag{29}$$

In conclusion, a DFT of size N has been transformed into two DFTs of size $N/2$. This technique can be recursively performed $m = log_2 N$ times for $N = 2^m$, resulting in a problem of order $O(Nlog_2 N)$. Each level of recursion adds a new butterfly matrix B resulting in a sequence of matrix multiplications (see Figure (8)). The butterfly matrices are represented with the letter B and the DFT block diagonal matrix with the letter F_B.

$$\hat{x}_N = \left(\prod_{k=1}^{\log_2 N} B_k \right) \cdot F_B \cdot \begin{bmatrix} x_{odd} \\ x_{even} \end{bmatrix} \tag{30}$$

In [25], Charles Van Loan suggests the following pseudocode for a recursive function where n has to be a power of two:

Algorithm 1 A Recursive Radix-2 FFT Procedure

Require: n is power of 2
Ensure: $y = F_n x$
  if $n = 1$ **then**
    $y \leftarrow x$
  else
    $m \leftarrow n/2$
    $\omega \leftarrow exp(-2\pi i/n)$
    $\Omega \leftarrow diag(1, \omega, \cdots, \omega^{m-1})$
    $z_T \leftarrow fft(x(0:2:n-1), m)$
    $z_B \leftarrow \Omega fft(x(0:2:n-1), m)$
    $y \leftarrow \begin{bmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{bmatrix} \begin{bmatrix} z_T \\ z_B \end{bmatrix}$
  end if

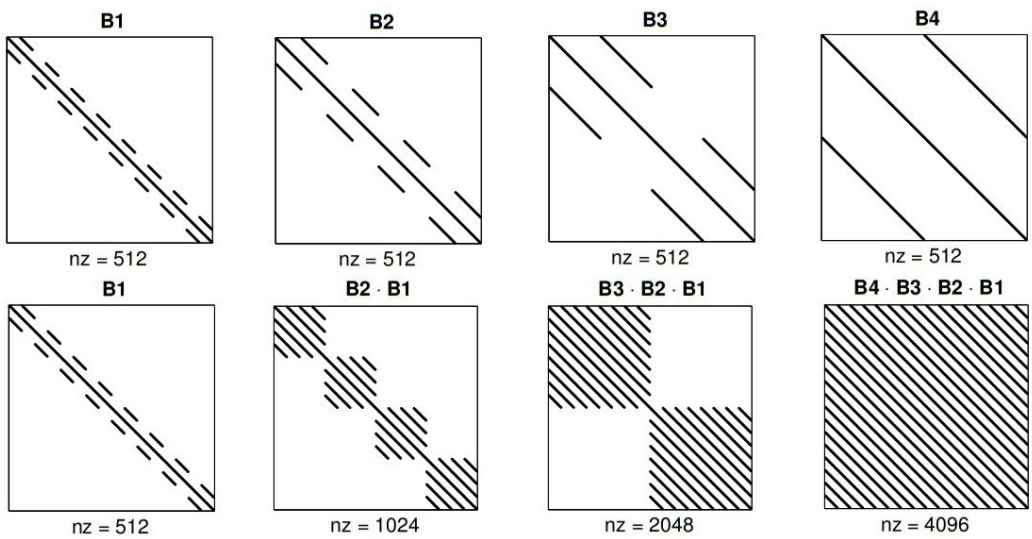


Figure 8 The single butterfly matrices B_i on the top; their products on the bottom. Source [22]

7.3.3 The Cooley-Tukey Radix-2 Factorization

The goal of this section is to represent the recursive algorithm from 7.3.2 in a non-recursive manner. Recursive algorithms are usually slower and use more memory than non-recursive algorithms.

The main idea of computing the FFT via matrix factorization is that the matrix F from equation (11) can be factored into a product of sparse matrices as can bee seen in equation (31) [25].

$$F_N = A_t \cdot A_{t-1} \cdots A_2 \cdot A_1 \cdot P_N^T, \tag{31}$$

$$A_q = diag(B_L, ..., B_L) = I_r \otimes B_L, \qquad L = 2^q, \quad r = N/L.$$

where the diagonal contains r elements. Matrix B_L can be obtained as follows:

$$B_L = \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix}$$

where $N = 2^t$, each A-matrix has 2 nonzeros per row and P_N^T is a permutation matrix. The A-matrix represents the bases of the transformation and can be calculated as a direct sum of the butterfly operators. The $\otimes$ symbol is the Kronecker product. This process can be easily computed using the following algorithm: [25]

Algorithm 2 A FFT Sparse Factorization Procedure

Require: $n = 2^t$
Ensure: $y = F_n x$
  $x \leftarrow P_n^T x$
  for $q = 1 : t$ **do**
    $x \leftarrow A_q x$
  end for
  $y \leftarrow x$

7.4 2D Fourier transform

As introduced before, any one-dimensional signal (for example, a sound wave) may be represented as a combination of one-dimensional frequencies. In this chapter, it will be explained how any two-dimensional signal (for example, an image) can be represented as a combination of two-dimensional frequencies.

Let $f(x, y)$ be a 2D function that has infinite support. The 2D Fourier transform is defined as:

$$\hat{f}(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx\, dy \tag{32}$$

where u and v are spatial frequencies perpendicular to each other.

The 2D inverse Fourier transform is defined as:

$$f(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}(u,v)e^{j2\pi(ux+vy)}du\,dv. \tag{33}$$

Some definitions that will be useful in future chapters are:

- $\hat{f}$ is a complex value that can be decomposed into $\hat{f} = \hat{f}_R + j\hat{f}_I$
- $|\hat{f}(u,v)|$ is the magnitude or amplitude spectrum.
- $arctan(\hat{f}_I/\hat{f}_R)$ is the phase angle spectrum

An image may be described as a two-dimensional function $f(x,y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x,y) is known as the image's intensity value at that location. We refer to a picture as a digital image when x, y, and the intensity values of f are all finite, discrete numbers [19]. An image can be thought of as a surface where each pixel indicates the height of the point in the mesh (see Figure (9)). In computers, digital images are usually represented with the RGB color model, in which the primary colors of light (red, green, and blue) are combined in various ways to give a wide range of colors. The component values for each pixel can be represented with three numbers, one for each primary light color, between 0 and 255, the range that a single 8-bit byte can offer. High-end digital image equipment usually uses larger integer ranges for each primary color. In this project, all images will be converted into black and white images with a pixel range between 0 and 255, so there is only one 8-bit channel of information for each image.

Figure 9 Mesh of an image of a cat. (If the GIF does not display, go to section A). Image source: [14]. Source own.

The 2D DFT and iDFT of a $f(n,m)$ matrix with N columns and M rows can be defined as:

$$\hat{f}(u,v) = \frac{1}{MN} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(n,m) e^{-j2\pi(\frac{ux}{N}+\frac{vy}{M})}, \tag{34}$$

$$f(n,m) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \hat{f}(u,v) e^{j2\pi(\frac{ux}{N}+\frac{vy}{M})}. \tag{35}$$

The essential concept is that the two-dimensional Fourier transform of a matrix is obtained by first applying the one-dimensional Fourier transform to each row of the matrix, and then applying the one-dimensional Fourier transform to each column of the intermediate matrix. FFT methods explained previously can be used to perform the one-dimensional Fourier transforms. Changing the order in which the rows and columns are Fourier transformed does not affect the outcome. The process of a 2D Fourier transform can be seen in figure 10.

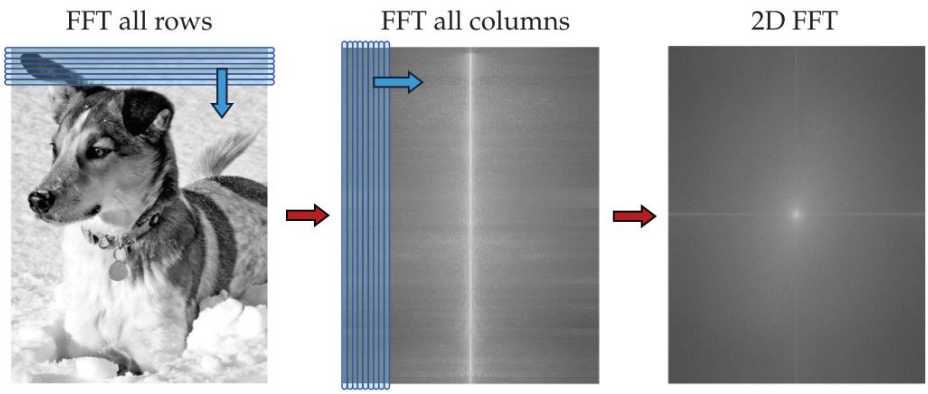


Figure 10 Schematic of 2D FFT. First, the FFT is taken of each row, and then the FFT is taken of each column of the resulting transformed matrix. Source: [5]

To better visualize the magnitude spectrum, the Fourier transform of the image is rearranged by shifting the zero-frequency component to the array's center. This is accomplished by swapping the first and third quadrant of the Fourier transform result, and the second and forth quadrant.

Interpreting the results obtained is no easy task but some patterns can be identified. The most obvious pattern that stands out most in almost all results is the bright dot in the center, where most of the energy in the Fourier domain is concentrated. The single pixel in the center of the image is that standard images have integer values between 0 and 255, hence the average is not zero. There is a DC component in the image known as the brightness of the image. The nearby values correspond to low-frequency data in the image domain. This corresponds to the images' many gradual changes. The second pattern that can be identified in almost all 2D Fourier transform is a bright cross in the center of the magnitude spectrum plot. The bright vertical and horizontal lines correspond to horizontal and vertical lines respectively in the image domain. These can be caused by the edges of the image due to its non-periodicity

In Figures (11) and (12) it can be seen how the frequency of the lines in the space domain determines the location of the energy in the amplitude spectrum. The higher the frequency of

the lines in the space domain, the farther away from the center is the energy in the amplitude spectrum. In both cases the phase spectrum is hard to read and 'seems' random. In Figure (13) it can be seen the amplitude and phase spectrum of lines with different frequencies at a 45-degree angle. The resulting amplitude spectrum is similar to the previous figures but now the phase spectrum has some patterns. In Figure (14) we can see how the phase spectrum changes with lines at different angles. These patterns are hard to predict unlike the patterns in the amplitude spectrum. In Figure (15) it can be seen how the amplitude spectrum remains almost constant with the change in location of the dot in the space domain, whereas the phase spectrum varies in each position. Therefore both, amplitude and phase spectrum are important, and it is essential to save and work with the real and imaginary parts of the results of the Fourier transform.

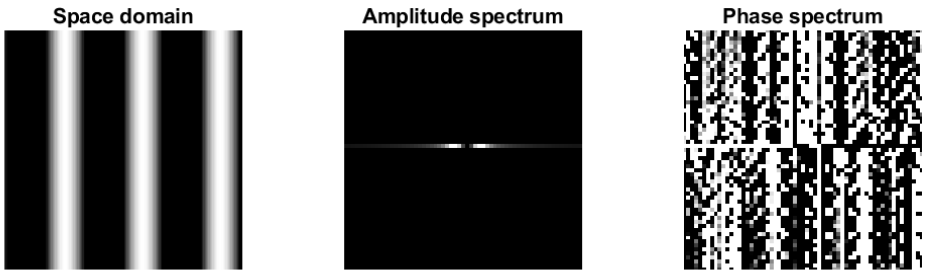


Figure 11 Vertical lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own.

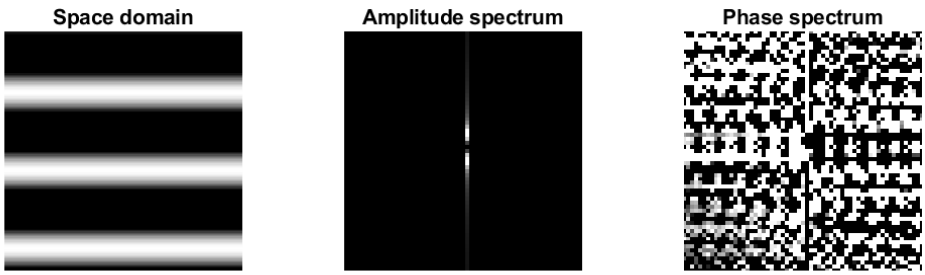


Figure 12 Horizontal lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own.

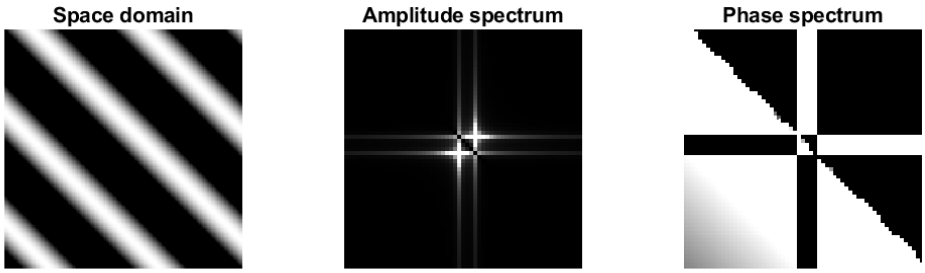


Figure 13 Diagonal lines and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own.

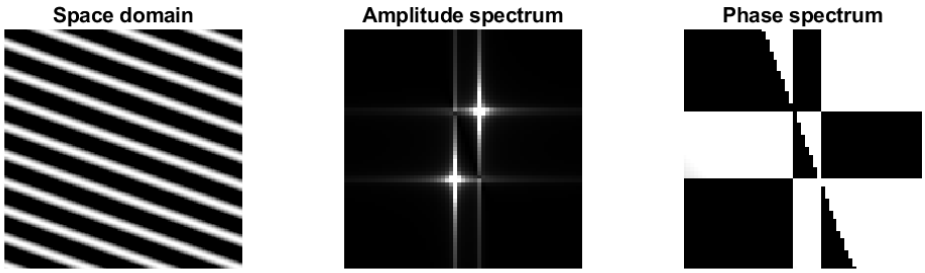


Figure 14 Lines rotating clockwise and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own.

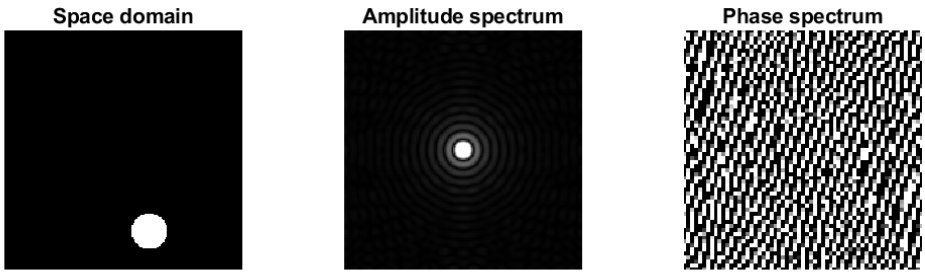


Figure 15 Moving dot and and its representation in the amplitude and phase spectrum. (If the GIF does not display, go to section A). Source own.

Now that we understand how the amplitude spectrum relates to the space domain, some filters can be applied to the images. A low pass filter can be performed by reconstructing an image in which all values that are far away from the center of the magnitude spectrum plot have been turned into zero. Analogously, for a high pass filter, all values near the center are turned into zero (see Figure (16)). This concept of applying filters to an image will be used for filtering compression in section 7.6.1 and for multilevel image compression in section 8.3.6.

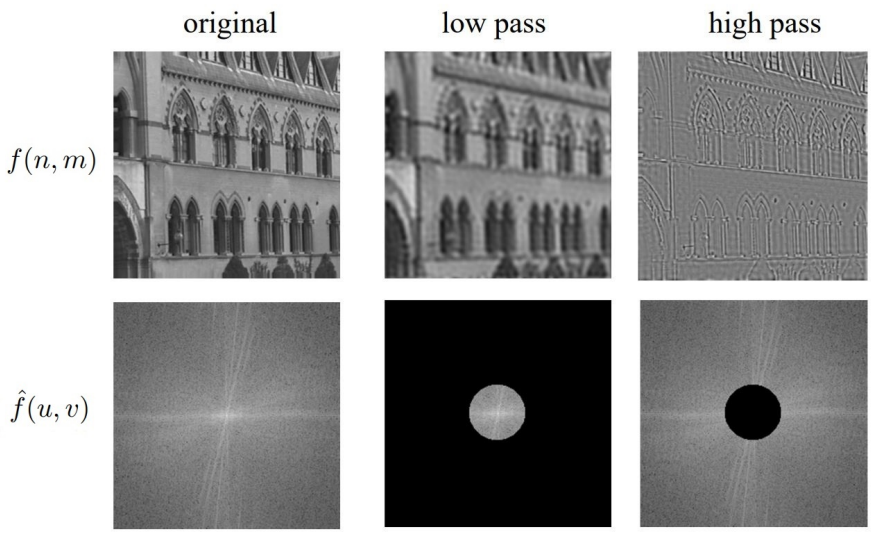


Figure 16 Action of low and high pass filters on a image. Source: [39]

7.5 Discrete cosine transform (DCT)

The DCT algorithm is one of the most widely used image compression algorithms. The international standard image compression technique JPEG is based on the DCT. Ahmed, Natarajan, and Rao initially introduced it in 1974 [3]. The DCT is a transform that is related to the DFT but only operates with real numbers [23]. We recall that the DFT is defined as:

$$\hat{x}(k) = \sum_{j=0}^{n-1} x(j) e^{-\frac{i2\pi jk}{n}} \qquad k = 0, 1, ..., n-1.$$

Applying some special symmetries to the input, the complex values of the DFT cancel out, which is very suitable for computation. There are various definitions of DCT depending on the periodicity and the type of symmetry applied to the original samples. The simplest definition is the following:

$$DCT(x) = Re[\hat{x}(k)] = \sum_{j=0}^{n-1} x(j) \cos\left(\frac{k2\pi j}{n}\right) \qquad k = 0, 1, ..., n-1. \tag{36}$$

The two-dimensional DCT of an M-by-N matrix A is defined as follows [13].

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\left(\frac{\pi(2m+1)p}{2M}\right) \cos\left(\frac{\pi(2n+1)q}{2N}\right), \tag{37}$$

$$0 \leq p \leq M-1 \qquad 0 \leq q \leq N-1,$$

$$\alpha_p = \begin{cases} 1/\sqrt{M} & p = 0 \\ \sqrt{2/M} & 1 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} 1/\sqrt{N} & q = 0 \\ \sqrt{2/N} & 1 \leq q \leq N-1. \end{cases}$$

The values B_{pq} are called the DCT coefficients of A.

Similarly, the inverse discrete cosine transform (iDCT) is given by:

$$A_{mn} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \alpha_p \alpha_q B_{pqn} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}, \tag{38}$$

$$0 \leq m \leq M-1 \qquad 0 \leq n \leq N-1.$$

According to the iDCT equation, any M-by-N matrix A may be expressed as a sum of MN functions, which are referred as the DCT's basis functions. The DCT coefficients B_{pq} are then the weights given to each basis function. Figure (19), (20), (21) and (22) show the Basis Functions of an 2-by-2, 4-by-4, 8-by-8 and 16-by-16 matrix. Vertical frequencies increase from top to bottom, whereas horizontal frequencies increase from left to right. The constant-valued basis function in the upper left is frequently referred to as the DC basis function, and the associated

DCT coefficient B_{00} is sometimes referred to as the DC coefficient.

These basis functions are the bridges between the frequency domain and the space domain, in other words, are what relate the frequency coefficients to the image. Each one of the basis functions is the base for one of the pixels in the DCT amplitude spectrum. The value of the DCT coefficient is just how much similar is the image of the study and the base function, therefore is just the weight. In Figures (23) and (24) the DCT of the DCT basis functions has been performed. It can be seen how the position of the pixel representing each basis function occupies the same relative position on the amplitude spectrum as the position of the basis function.

The first DCT method for image compression is to use the entire image of the study as input (see Figure (17)). Unlike in the Fourier transform, the amplitude spectrum in the DCT does not require a shift, so low frequencies appear in the top left corner. This method archives a really good compression and reconstruction but has some problems. Not all images have the same number of pixels, so the basis functions will change from image to image. Consequently, every time we desire to compress or reconstruct an image, the basis function should be calculated first, which is time-consuming and requires computation resources.

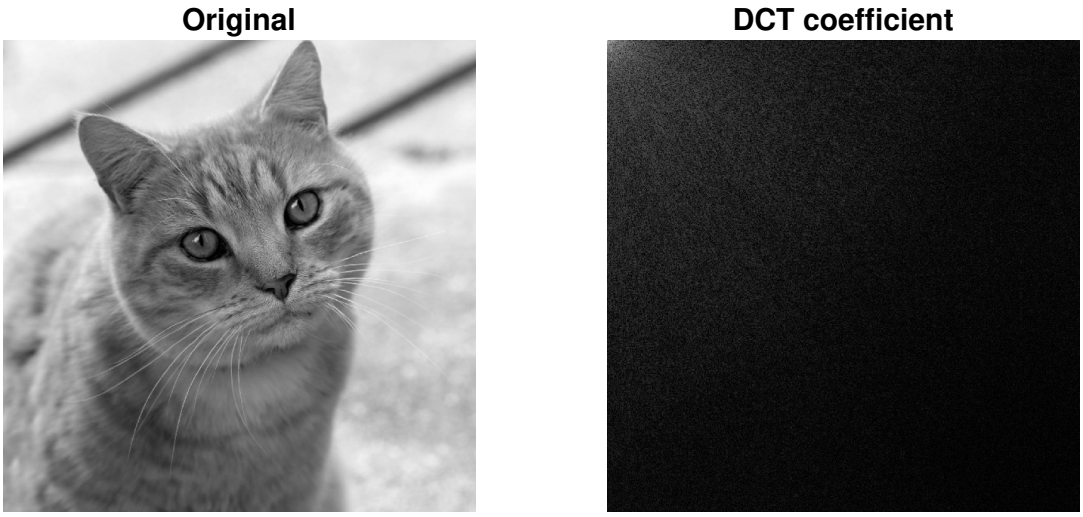


Figure 17 An image and its DCT coefficients. Source own

To solve this problem, we can break it down into smaller problems. The input image can be divided into smaller block images, typically 8-by-8 pixel images [2], which can be processed independently. The block division can be seen in Figure (18). This method solves the problem of having to calculate the basis functions every time for each image. With normalized blocks, the basis functions are now data that every computer can store and use efficiently. This is the method that was introduced in 1992 by the Joint Photographic Experts Group and CCITT (now ITU-T), the widely used JPEG. [8]

[2]The 8-by-8 block structure was chosen because it was the maximum block-size that integrated circuit technology could handle at the time the standard JPEG was created. [34]

The main reason why the DCT can obtain better results than the FT for image compression is that the DCT has one-half cycle basis functions. This are the basis function right next to the DC coefficient seen in Figures (19), (20), (21) and (22). The slopes of these basic functions are gentle from one side of the array to the other. On the other hand, the lowest frequencies in the Fourier transform construct one complete cycle. Almost all images feature areas where the brightness gradually changes over a region. Better compression can be achieved by using a basis function that matches this fundamental pattern.

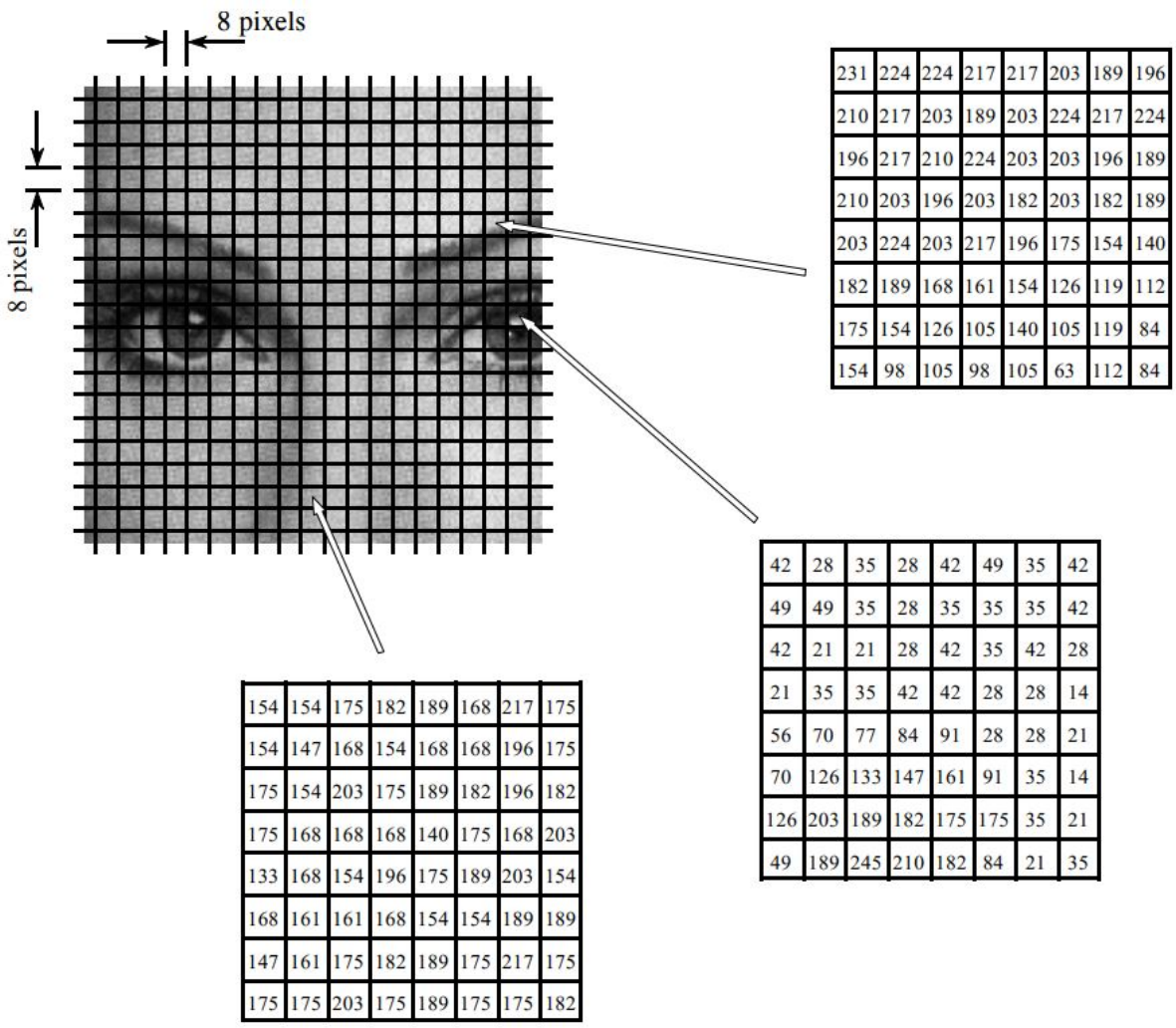


Figure 18 JPEG image division. JPEG transform compression starts by breaking the image into 8×8 groups, each containing 64 pixels. Three of these 8×8 groups are enlarged in this figure, showing the values of the individual pixels, a single byte value between 0 and 255. Source [34]

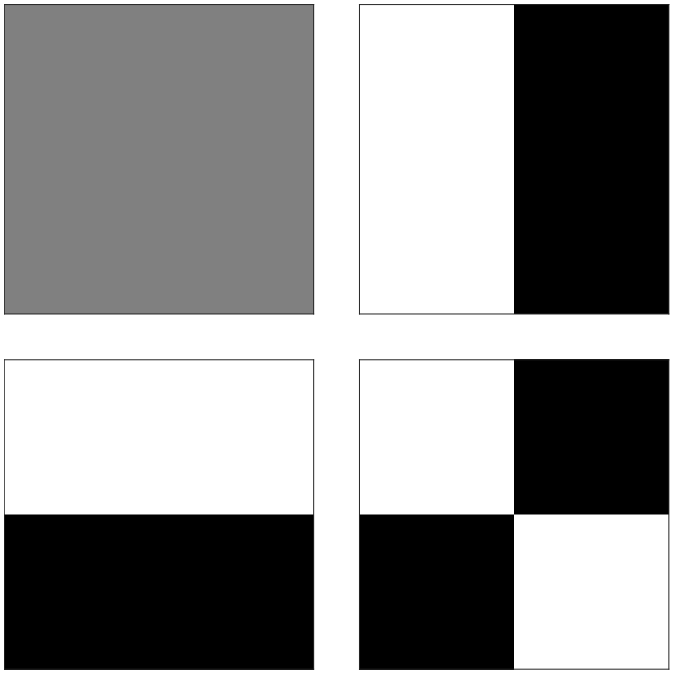


Figure 19 The 4 DCT Basis Functions of an 2-by-2 matrix. Source own.

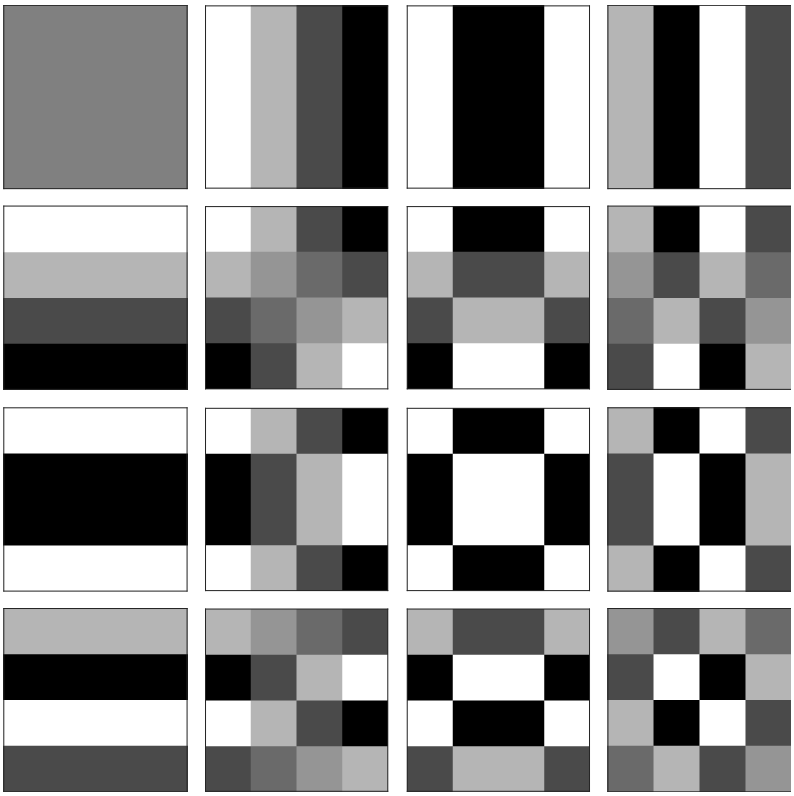


Figure 20 The 14 DCT Basis Functions of an 4-by-4 matrix. Source own.

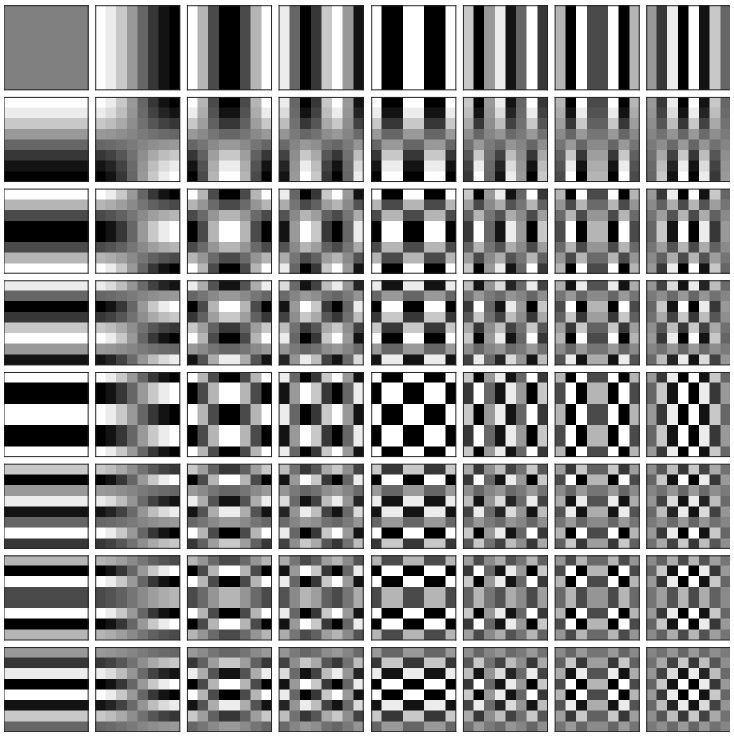


Figure 21 The 64 DCT Basis Functions of an 8-by-8 matrix. Source own.

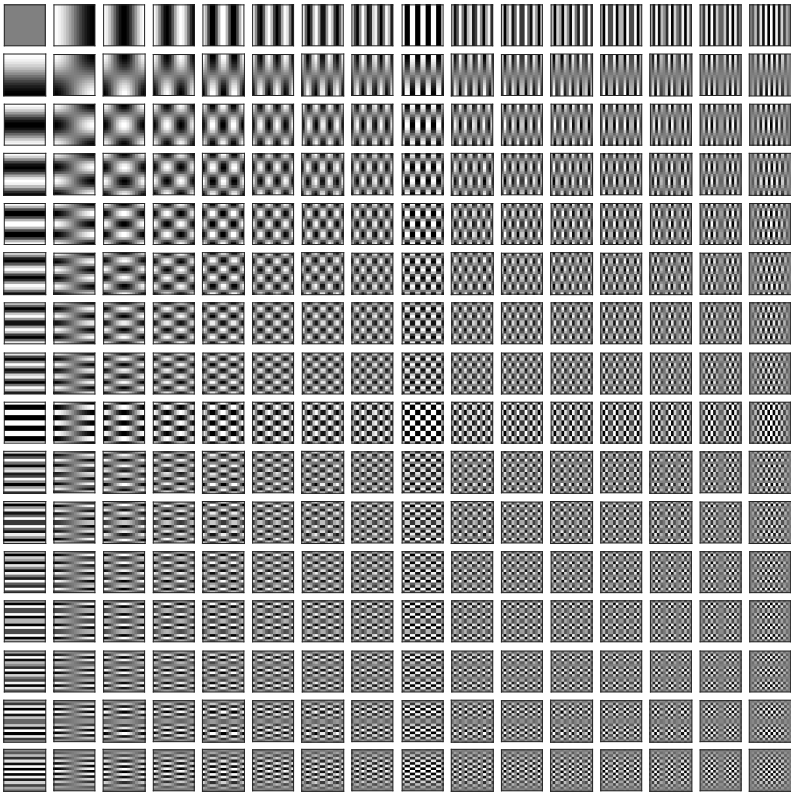


Figure 22 The 256 DCT Basis Functions of an 16-by-16 matrix. Source own.

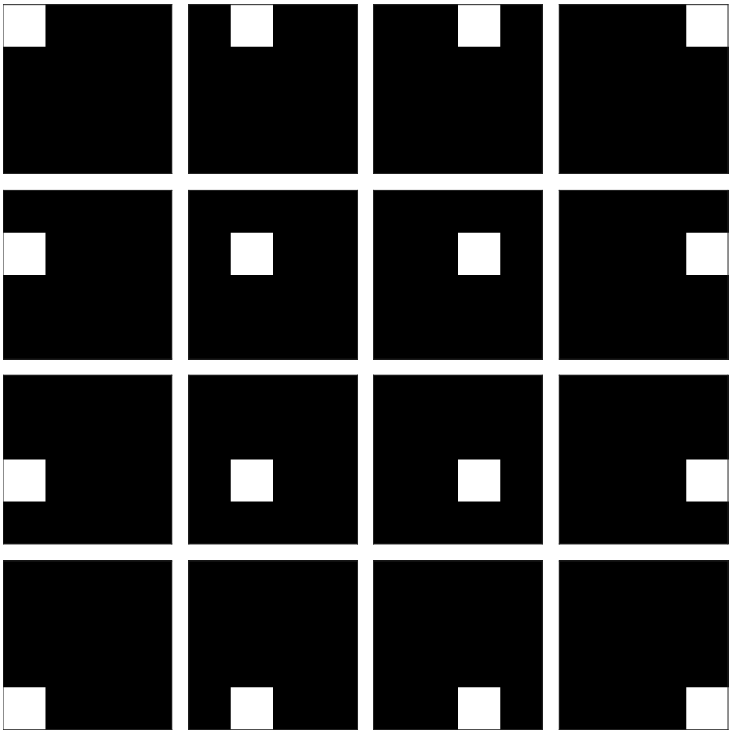


Figure 23 The DCT of the 14 DCT Basis Functions of an 4-by-4 matrix. Source own.

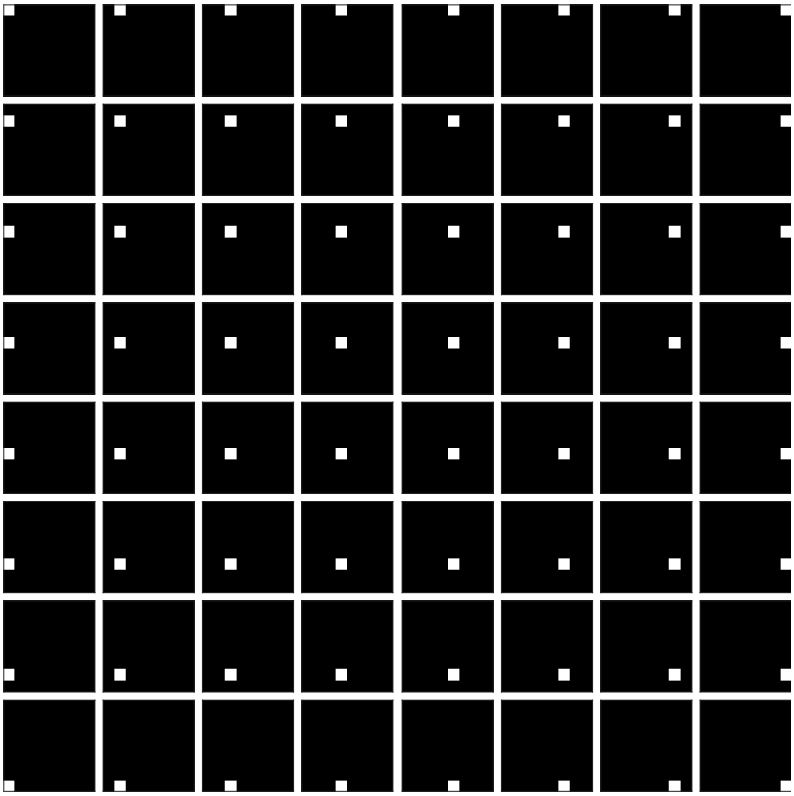


Figure 24 The DCT of the 64 DCT Basis Functions of an 8-by-8 matrix. Source own.

7.6 Image compression using Fourier transform

The main objective of this project is to perform image compression. So far, everything seen in previous sections has been about how to transform information from a temporal-spatial domain to a frequency domain. This transformation has been carried out because in the frequency domain not all information has the same weight so some values can be removed. There are several compression methods all of which take advantage of this property. We will study three of them: filtering, thresholding, and JPEG.

7.6.1 Filtering compression

Low-frequency signal components are more significant than high-frequency signal components. Only 5% of the encoded information would be lost if 50% of the bits from the high-frequency components were removed [34]. This means the highest frequency components can be eliminated with a low-pass filter, while only degrading the signal a small amount (see Figures (25) and (26)).

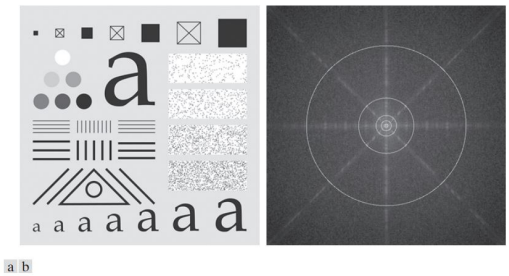


Figure 25 (a) Test pattern of size 688 × 688 pixels, and (b) its spectrum. The spectrum is double the image size as a result of padding, but is shown half size to fit. The circles have radii of 10, 30, 60, 160, and 460 pixels with respect to the full-size spectrum. Source: [19]

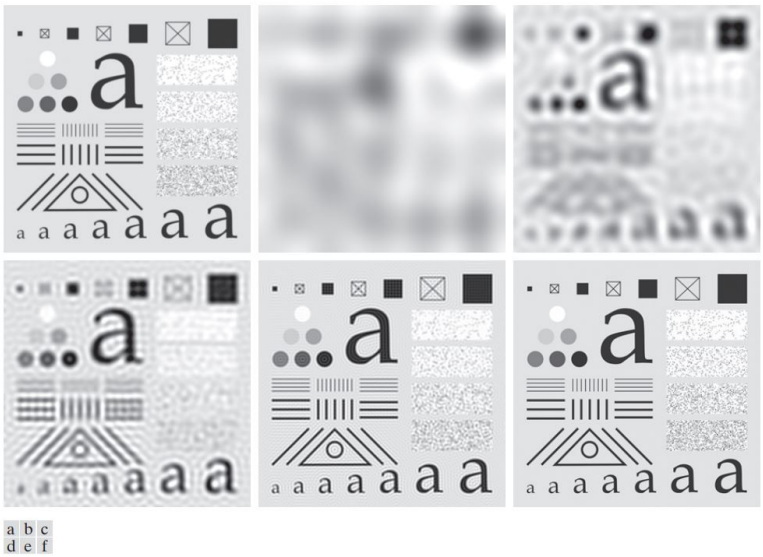


Figure 26 (a) Original image of size 688 × 688 pixels. (b)–(f) Results of filtering using ideal low pass filters with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Figure (25)(b). We used mirror padding to avoid the black borders characteristic of zero padding. Source [19]

7.6.2 Thresholding compression

Sometimes we are interested to save some of the high-frequency information so applying a low pass filter is not a viable solution. To make image compression more efficient, instead of defining a region, we can define a cutoff value for the weight of the coefficients where all values below it will be changed to zero. This method is called thresholding and it ensures that if any high frequency is important we won't get rid of it by doing the compression. In Figure 27 thresholding compression method has been performed for the same number of bits saved as in the case of Figure 26 for filtering compression.

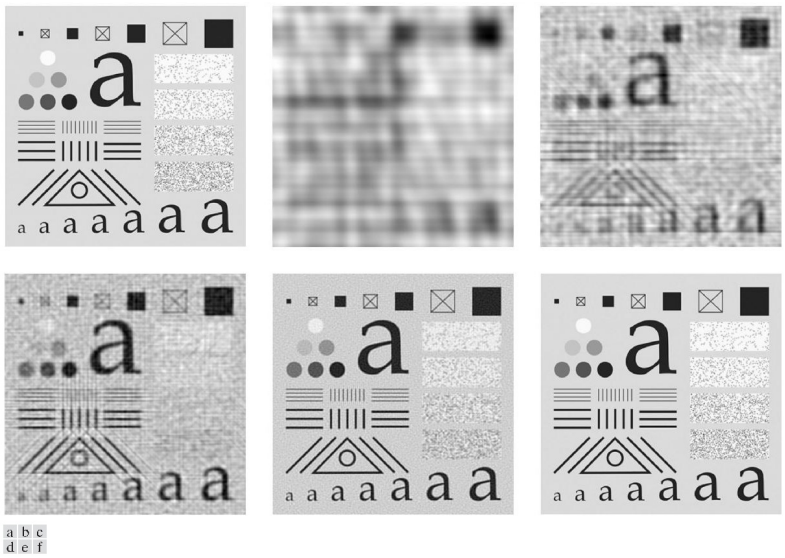


Figure 27 (a) Original image of size 688 × 688 pixels. (b)–(f) Results of threshold compression removing the same number of information as each corresponding case of Figure (26). We used mirror padding to avoid the black borders characteristic of zero padding. Original image form [19]. Source own.

This method has a problem. To be able to reconstruct the image we need not only the value of the coefficients but also their position in the plane. In the previous method where we defined a region, this was not a problem because all the values of the coefficients could be saved one after the other without having to save their positions since they can be deduced from their position in the vector. In the thresholding method, the coefficients do not have to be successive so this can't be done, therefore all the coefficients have to be saved and Run-Length encoding will be performed.

Run-Length encoding takes advantage of the same character repeated many times in a row, in our case will be all the zeros from thresholding. Two values are sent to the output file every time a zero is detected in the input data. The first of these numbers is a zero, which indicates the start of run-length compression and the second one indicates the number of zeros in the run. Compression will occur if the average run-length is greater than two. Many single zeros in the data, on the other hand, can make the encoded file bigger than the original. Figure (28) shows an example of run-length encoding.

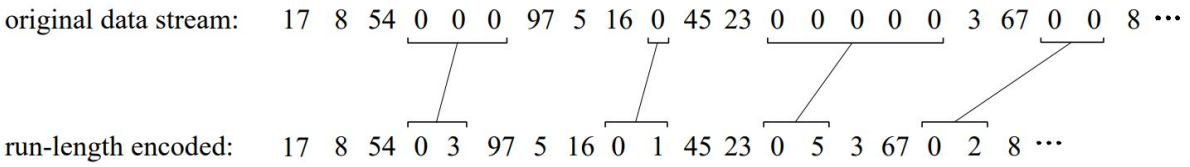


Figure 28 Example of run-length encoding. Each run of zeros is replaced by two characters in the compressed file: a zero to indicate that compression is occurring, followed by the number of zeros in the run. Source [34].

7.6.3 JPEG compression

JPEG is the image compression standard. This method exemplifies how multiple data compression schemes can be combined for improved performance. This method is described in *CCITT Recommendation T.81: Terminal Equipment and Protocols for Telematic Services: Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines*. As seen in chapter 7.5, breaking the image into 8-by-8 pixel groups and performing a DCT on each block is the first step in JPEG compression. Then, each 8×8 spectrum is compressed by reducing the number of bits and eliminating some of the components. This process is known as quantization and controls the level of compression. This step uses the values from a quantization table in the following way. Each value in the spectrum is divided by the matching value in the quantization table, and the result is rounded to the nearest integer. Two examples of quantization tables are shown in Figure (29). For example, in the low compression quantization table, the upper-left value is one, therefore the DC value will remain unaltered. In contrast, the lower-right item in the same table is 16, reducing the original range of the DCT spectrum coefficients from -127 to 127 to just -7 to 7. In other words, the precision of the value has been lowered from eight to four bits. In the high compression table, the lower-right value is 256, removing the spectral value entirely.

a. Low compression

1	1	1	1	1	2	2	4
1	1	1	1	1	2	2	4
1	1	1	1	2	2	2	4
1	1	1	1	2	2	4	8
1	1	2	2	2	2	4	8
2	2	2	2	2	4	8	8
2	2	2	4	4	8	8	16
4	4	4	4	8	8	16	16

b. High compression

1	2	4	8	16	32	64	128
2	4	4	8	16	32	64	128
4	4	8	16	32	64	128	128
8	8	16	32	64	128	128	256
16	16	32	64	128	128	256	256
32	32	64	128	128	256	256	256
64	64	128	128	256	256	256	256
128	128	128	256	256	256	256	256

Figure 29 JPEG quantization tables. These are two example quantization tables that might be used during compression. Each value in the DCT spectrum is divided by the corresponding value in the quantization table, and the result rounded to the nearest integer. Source [34].

After that, the DCT spectrum coefficients are converted into a linear sequence of 64 values using the serpentine pattern described in Figure (30). This groups all of the high-frequency components together, allowing run-length encoding to efficiently compress the vast number of zeros. In addition, the coefficients are sent through an entropy encoding technique, which compresses the data even further. There are several entropy coding procedures but the one used the most is the Huffman coding. This entropy coding technique won't be studied in this project.

A simplified diagram of the JPEG compression algorithm can be seen in Figure (31). In this diagram, the run-length encoding is not displayed because is not always executed. The run-length encoding is performed right after the quantizer and before the entropy encoder.

In the code developed in this thesis, neither quantization nor entropy encoding nor run-length encoding will be performed.

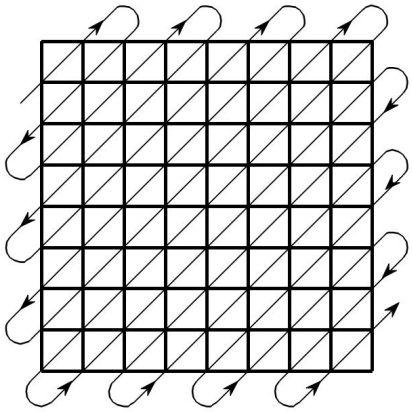


Figure 30 Serpentine pattern used in JPEG compression to convert the 8×8 DCT spectrum into a linear sequence of 64 values. Source [34].

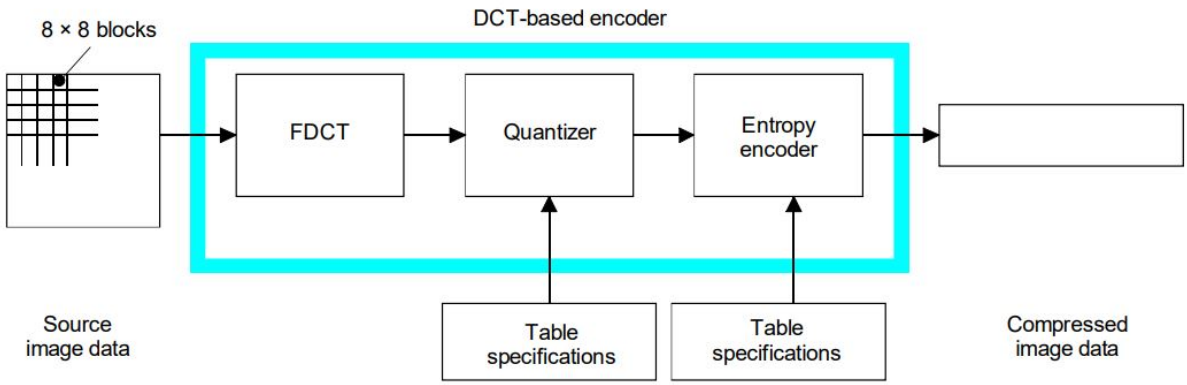


Figure 31 JPEG DCT-based encoder simplified diagram. Source [8].

7.7 Short-Time Fourier Transform

While the Fourier transform provides detailed information about a signal's frequency content, it does not provide any information about when those frequencies occur in time. Only truly periodic and stationary signals can be described using the Fourier transform. It is critical to simultaneously characterize the frequency content and its evolution in time for a signal with nonstationary frequency content, such as the audio signal of musical composition. Most data sets have a time (or space) and frequency correlation.

To overcome the Fourier transform's limitations, Dennis Gabor introduced the short-time Fourier transform (STFT) in his paper "Theory of communication," published in 1946. The STFT computes a windowed FFT in a moving window (see Figure (32)) [5, 18, 17]. This STFT allows frequency content to be localized in time, resulting in a spectrogram, a plot of frequency versus time. The spectrograms of various audio signals are shown in Figure (35), and it can be seen how we obtain both frequency and time information. In the case of the spectrogram of a bird chirp, we can see how the frequency of the signal varies with time and that it pulsates. In the case of the train whistle, we can see both harmonics and audio intensity over time.

The shot-time Fourier transform of a signal $f(t)$ with a translating Gaussian window is defined by this formula:

$$STFT(\tau, f) = \langle x, g_{\tau,f} \rangle \int_{-\infty}^{\infty} x(t) g^*_{\tau,f}(t) dt = \int_{-\infty}^{\infty} x(t) g(t-\tau) e^{-i2\pi ft} dt \tag{39}$$

where $g_{\tau,f}(t)$ is defined as

$$g_{\tau,f}(t) = e^{i2\pi ft} g(t-\tau). \tag{40}$$

The function $g(t)$ is the kernel, and is often chosen to be a Gaussian:

$$g(t) = e^{-(\tau-t)^2/a^2} \tag{41}$$

where the time window is scaled by the parameter a and the traslation in time is controled by the parameter τ. See Figure (33).

The inverse STFT is given by:

$$f(t) = \frac{1}{2\pi \|g\|^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} STFT(\tau, f) g(\tau - t) e^{i2\pi ft} df dt. \tag{42}$$

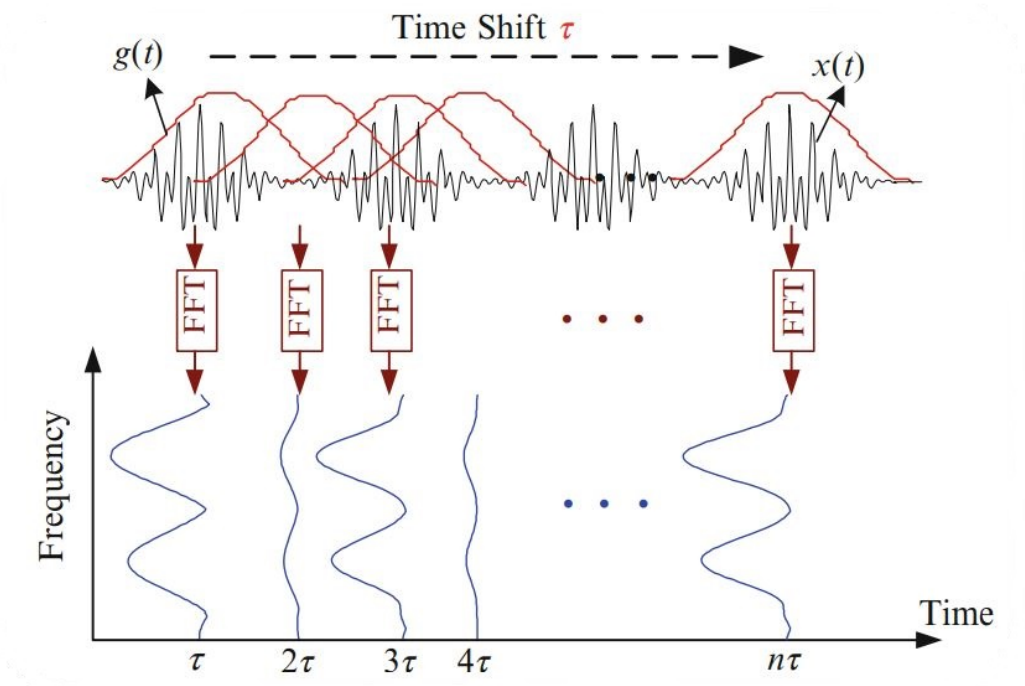


Figure 32 Illustration of short-time Fourier transform on the test signal $x(t)$ and a translating window $g(t)$. Source [18]

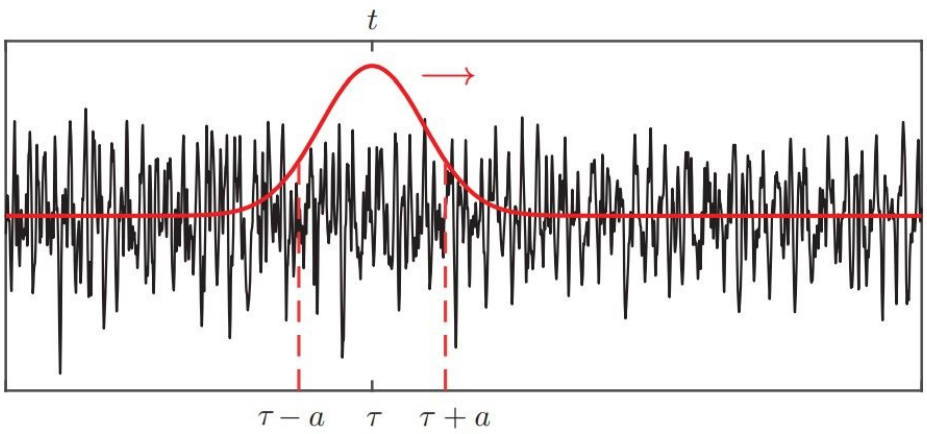


Figure 33 Illustration of the translating Gaussian window for the short-time Fourier transform. Source [5]

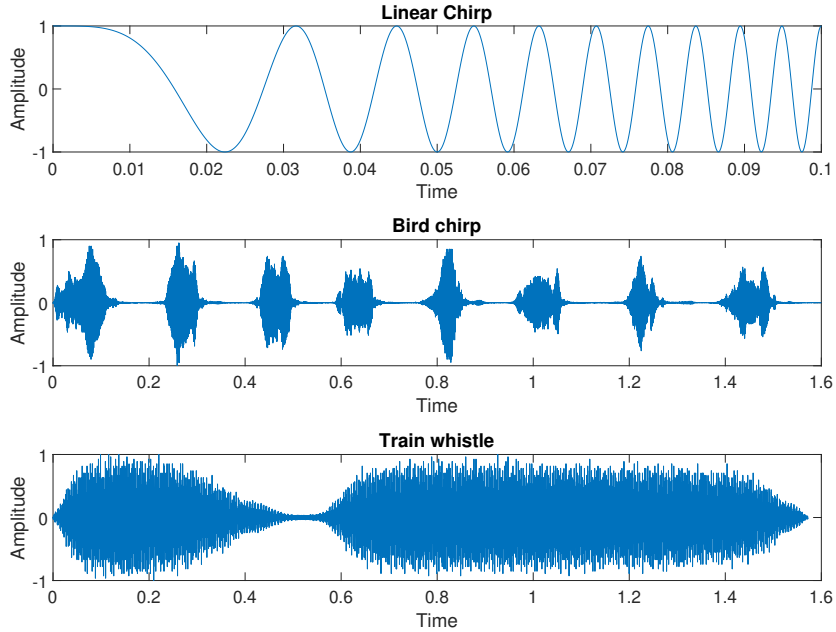


Figure 34 Audio signals. Source own.

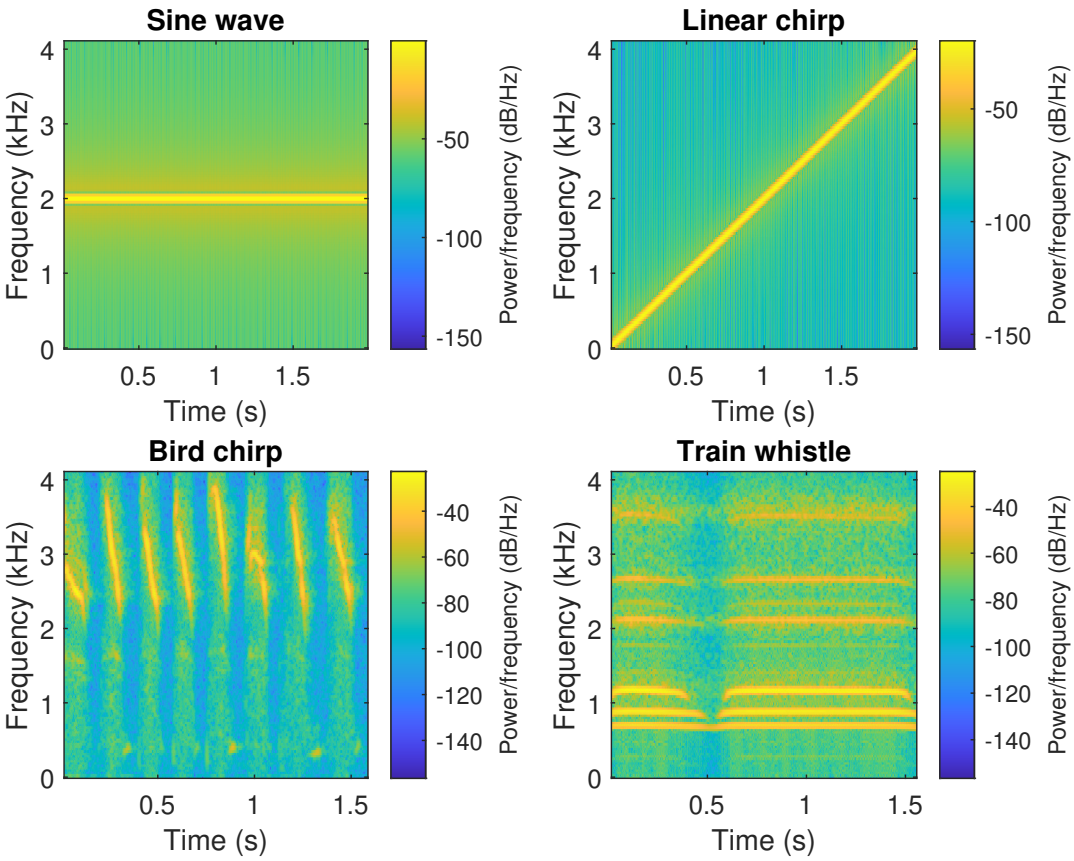


Figure 35 Examples of spectograms of a sine wave and signals from Figure (34). Source own.

7.8 Uncertainty Principle

There is a fundamental uncertainty principle in time-frequency analysis that limits the ability to achieve high resolution in both the time and frequency domains at the same time. A time series is perfectly resolved in time but not in frequency content, and the Fourier transform perfectly resolves frequency content but not in time. The spectrogram resolves both time and frequency information, but each domain has a lower resolution.

An alternative approach is a multi-resolution analysis. Because lower frequencies tend to last longer over time than higher frequencies, lower frequencies can be resolved to obtain the frequency content more precisely because they are not localized in time. Whereas higher frequencies can be resolved in smaller time domains to obtain the frequency content more localized in time.

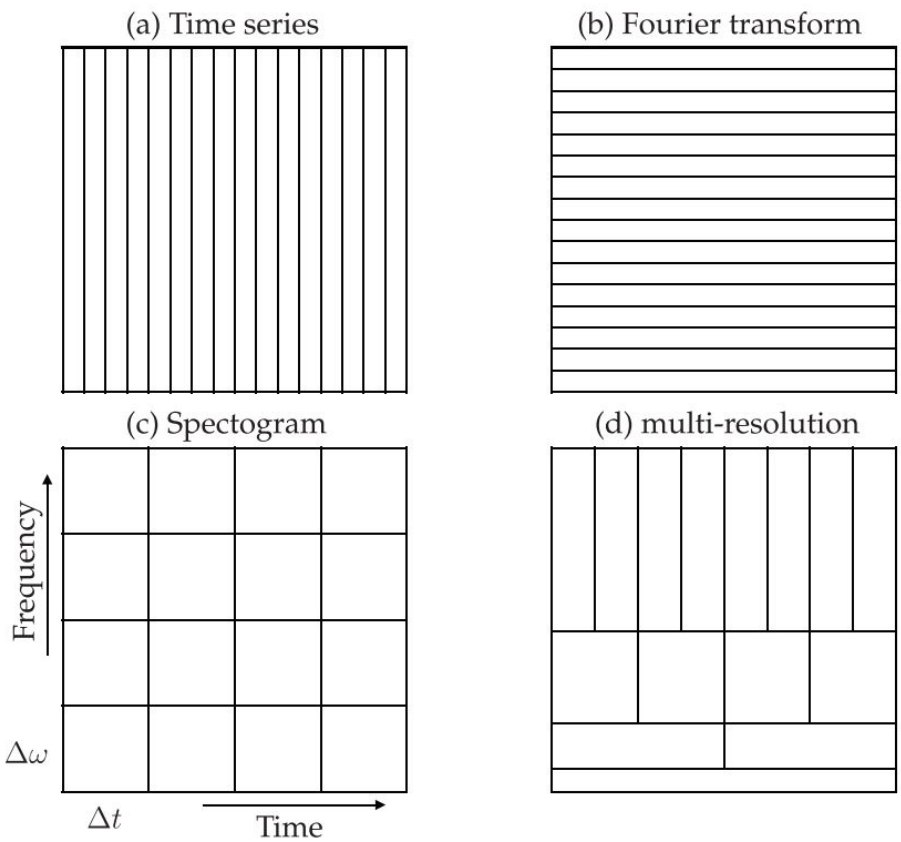


Figure 36 Illustration of resolution limitations and uncertainty in time-frequency analysis. Source: [5]

8 Wavelet transform

Wavelets adapt Fourier analysis techniques to more generic orthogonal bases, and by utilizing a multi-resolution decomposition, they can partially overcome the uncertainty principle described above. Its ability to examine a signal in both time and frequency in a way that is fundamentally different from the traditional short-time Fourier transform has withdrawn a lot of attention in the last few decades. The JPEG2000 image compression format uses the wavelet transform to perform its compression. This method will not be studied in as much depth as JPEG in section 7.6.3 due to its greater complexity.

8.1 Theoretical background

The basic concept behind wavelet analysis is to start with a function $\psi(t)$, which is referred to as the mother wavelet, construct a family of scaled and translated versions of the function, and calculate how much similar are this family of waves to our signal being studied. The normalized wavelet function can be defined as: [2, 5]

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{43}$$

where, like in the Gabor transform, the parameters a and b are responsible for scaling and translating the function respectively. When the value of a is reduced, the wavelet becomes squished. As a result, high-frequency data can be captured. On the other hand, increasing the value of a stretches the wavelet and captures low-frequency information. By lowering b, the wavelet will shift to the left and by increasing b, it will shift to the right. In Figure (37) it can be see the dilation and translation of a wavelet.

The length of the time domain is related to the distribution of energy in the spectral domain. The effect of dilation on the corresponding energy spectrum of a Mexican hat wavelet is shown in Figure (38). The energy spectrum associated with the wavelet contracts in a smaller region as the wavelet in the time domain expands. This is an obvious consequence, as time domain expansion must imply a lengthening of time period and a corresponding decrease in associated frequencies as seen before in the Fourier chapter.

Paul Addison in his book *The Illustrated Wavelet Transform Handbook* defines the wavelet as a "mathematical microscope, where b is the location on the time series being viewed and a is associated with the magnification at location b". [2]

If these functions are orthogonal then the basis can be brought into play for projection, as in the Fourier transform. The basic concept is to calculate how much of a wavelet is present in a signal at a given scale and location. This can be calculated by computing the convolution of the signal with the wavelet at various scales. One thing to keep in mind is that, unlike the Fourier transform, which uses just the sine and cosine functions, wavelet transforms do not have a single set of basis functions. Wavelet transformations, on the other hand, have an endless number of potential basis functions. As a result, wavelet analysis enables instant access to information that other time-frequency approaches, such as Fourier analysis, might hide. [20]

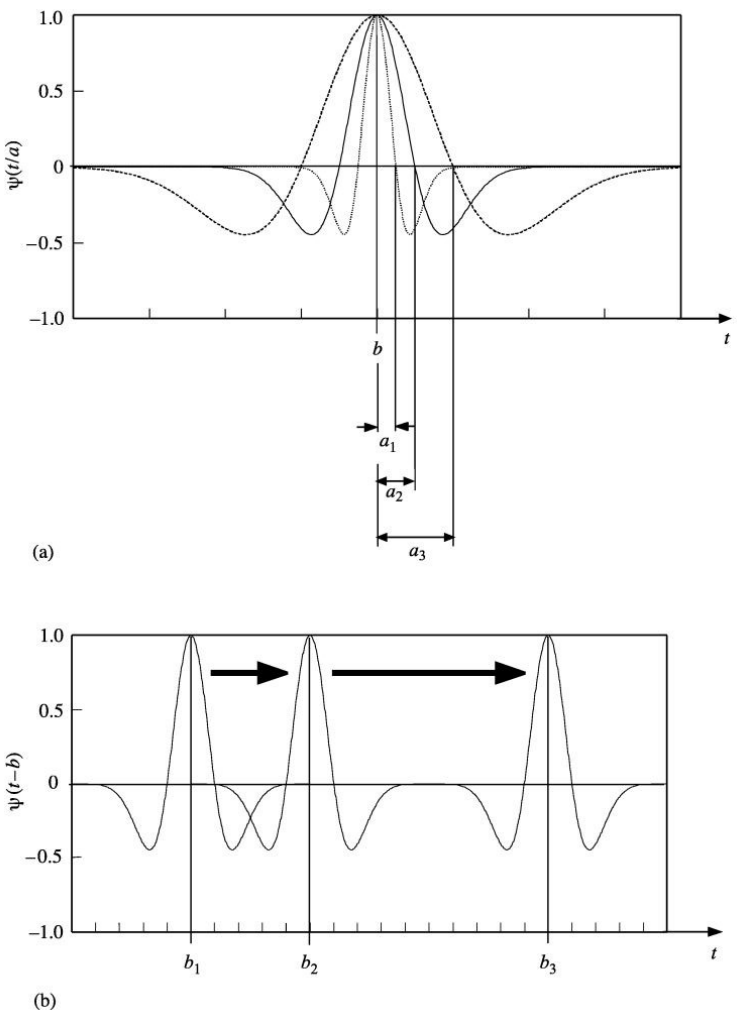


Figure 37 Dilation and translation of a wavelet. (a) Stretching and squeezing a wavelet: dilation ($a_1 = a_2/2; a_3 = 2 \cdot a_2$). (b)Moving a wavelet: translation. Source: [2]

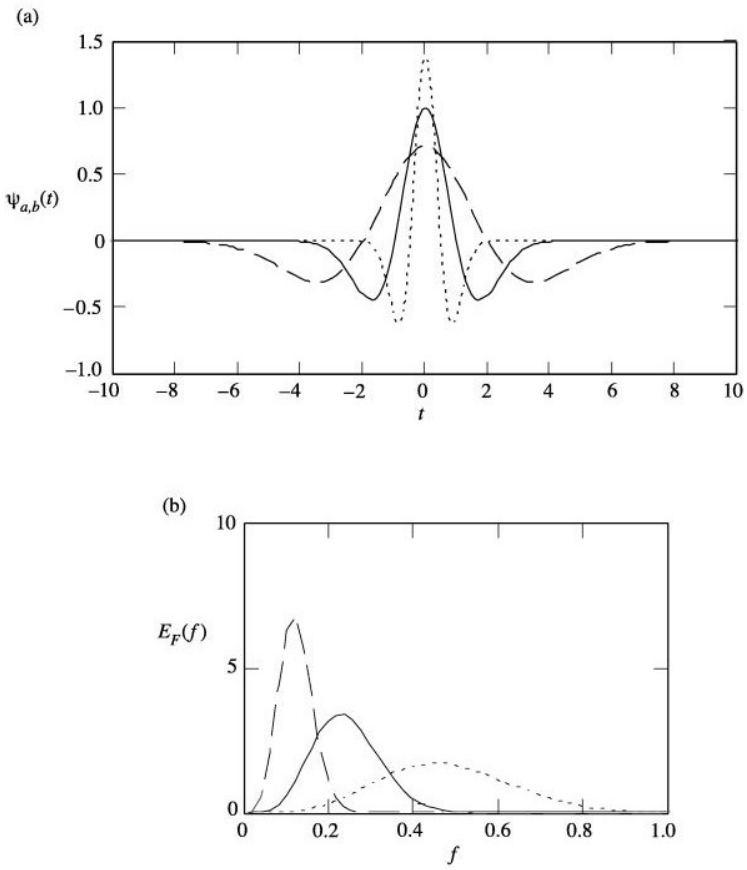


Figure 38 The Fourier energy spectra of dilating wavelets. (a) Thre Mexican hat wavelets at three dilations, a = 0.5, 1.0, 2.0 and all located at b = 0. (b)Energy spectra corresponding to the wavelets in (a) all have identical energy and hence the same area under their curves. Note that only the positive part of the energy spectrum is shown. Source: [2]

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

There are a variety of wavelets to choose from depending on the type of signal and the characteristic information that needs to be extracted. Some of the most used wavelets are shown in figure 39. A fractal structure may be found in several wavelet bases. One example is the Daubechies wavelet family (see figure 40). The JPEG2000 format uses the CDF 9/7 wavelet to perform compression.[3]

A function $\psi(t)$ must fulfill certain mathematical criteria in order to be classified as a wavelet: [1, 2]

1. A wavelet must have finite energy:

$$E = \int_{-\infty}^{+\infty} |\psi(t)|^2 dt < \infty \tag{44}$$

   where the vertical brackets represent the modulus operator which gives the magnitude of $\psi(t)$.

2. Admissibility condition:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(f)|^2}{|f|} df < \infty \tag{45}$$

   where $\hat{\psi}(f)$ is the Fourier transform of $\psi(t)$. This condition implies that the wavelet does not have a zero-frequency component, $\hat{\psi}(0) = 0$. C_ψ is known as the admissibility constant, and its value is determined by the wavelet used.

3. For complex wavelets an additional criterion must be met. This criterion is not of special interest for this study. This criterion is explained by Paul Addison in *The Illustrated Wavelet Transform Handbook* page 10.

The Haar wavelet is the simplest and earliest example of a wavelet (Figure 39 c.) In figure 41 three Haar wavelets are shown ($\psi_{1,0}$,$\psi_{1/2,0}$ and $\psi_{1/2,1/2}$) representing the bases for the first two levels of the multi-resolution analysis. The resulting Haar wavelets are orthogonal, providing a hierarchical basis for a signal since each higher frequency layer is a bisection of the next layer down.

$$Haar: \qquad \psi(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{46}$$

[3]Cohen–Daubechies–Feauveau wavelets (CDF) are a family of biorthogonal wavelets that was made popular by Ingrid Daubechies.

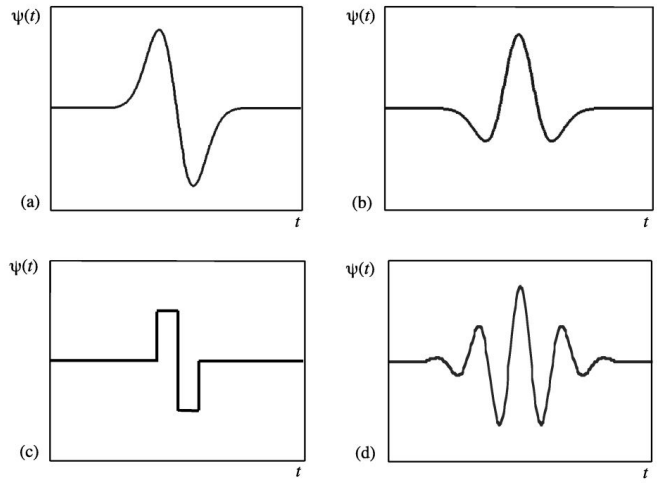


Figure 39 Four wavelets. (a)Gaussian wave(first derivative of a Gaussian). (b)Mexican hat (second derivative of a Gaussian). (c)Haar. (d)Morlet (real part). Source: [2]

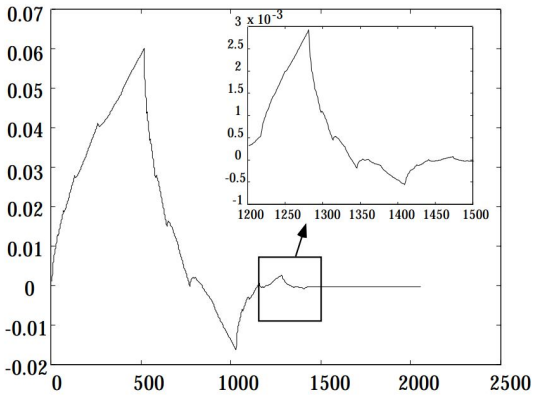


Figure 40 The fractal self-similiarity of the Daubechies mother wavelet. Source: [20]

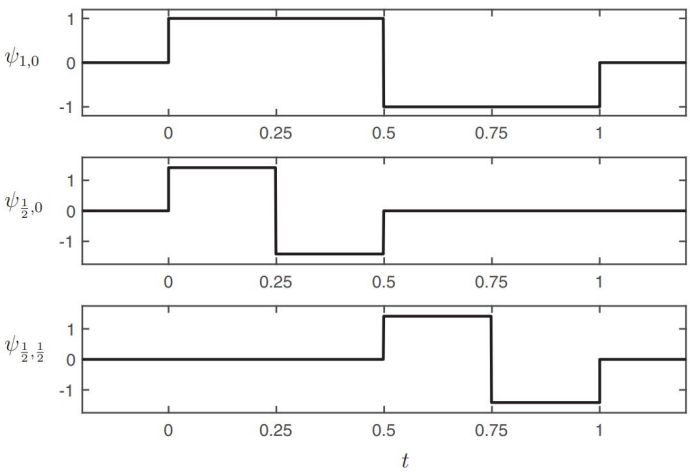


Figure 41 Three Haar wavelets for the first two levels of the multi-resolution. Source: [5]

8.2 Continuous Wavelet Transform

The continuous wavelet transform (CWT), like the Fourier transform, employs inner products to quantify the similarity between a signal and an analysis function. The analyzing functions in the Fourier transform are complex exponentials, $e^{-i2\pi ft}$ and the resultant transform is a function of a single variable, f. The short-time Fourier transform quantifies the similarity between the signal and its analysis signal in a given length interval centered at τ. The analytical functions in the short-time Fourier transform are windowed complex exponentials, $g(t-\tau)e^{-i2\pi ft}$, and the output is a two-variable function, $STFT(\tau, f)$. Now, the analyzing function in the CWT are wavelet versions of a mother wavelet that has been moved, compressed, or stretched, $\psi_{(a,b)}$. The continuous wavelet transform (CWT) can be defined as:

$$\mathcal{W}_\psi(x)(a,b) = \langle x, \psi(a,b)\rangle = \int_{-\infty}^{\infty} x(t)\psi^*_{a,b}(t)dt \tag{47}$$

where $\psi^*_{a,b}$ stands for the complex conjugate of $\psi_{a,b}$.

The inverse continious wavelet transform (iCWT) is defined as

$$x(t) = \frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{\infty} \mathcal{W}_\psi(x)(a,b)\psi_{a,b}(t)\frac{1}{a^2}db\,da. \tag{48}$$

The CWT is basically the multiplication of chosen wavelet with the signal being analyzed for different versions of the wavelet. In Figure (42) can be seen how for regions A and B where both, the signal and the wavelet, have the same sign, therefore the contribution is positive. Whereas in regions C, D, and E, the signal and wavelet have a different sign so the contribution is negative.

Figure (43) illustrates a basic sinusoidal waveform superposed with Mexican hat wavelets with different values for the parameters a and b. The value of the wavelet transform integral is determined by the wavelet's location and dilation.

CWTs are generally computed throughout a continuous range of a and b. The plot of $\mathcal{W}(a,b)$ versus a and b is known as a wavelet transform plot (see Figure (44)). In the left half of the plot, the not near-zero values of $\mathcal{W}(a,b)$ are at large values of a. Whereas for the right half of the plot is the other way around, not near-zero values of $\mathcal{W}(a,b)$ are at small values of a. In both cases, when the frequency of the wavelet matches the frequency of the signal, substantial undulations appear in $\mathcal{W}(a,b)$ corresponding to the sinusoidal form of the signal. These large undulations occur when the wavelet is similar in size to the undulations in the signal being studied. The undulations occur because the wavelet moves in and out of phase with the signal as it is translated in time by the parameter b. These undulations can be observed in a surface plot in Figure (45).

The CWT is a method, similar to STFT, that can extract frequency information without sacrificing all temporal information. Two adjustments are made to better show the wavelet transform plot and convey the results more intuitively. The first is that the y-axis is reversed, with small a values on top and large values on the bottom. This corresponds to the frequency distribution. The second change is to represent the absolute values of $\mathcal{W}(a,b)$ so that the undulations are not as visible. In Figures (46) and (47) it can be seen the CWT plot for a sinusoidal and a linear chirp signal respectively.

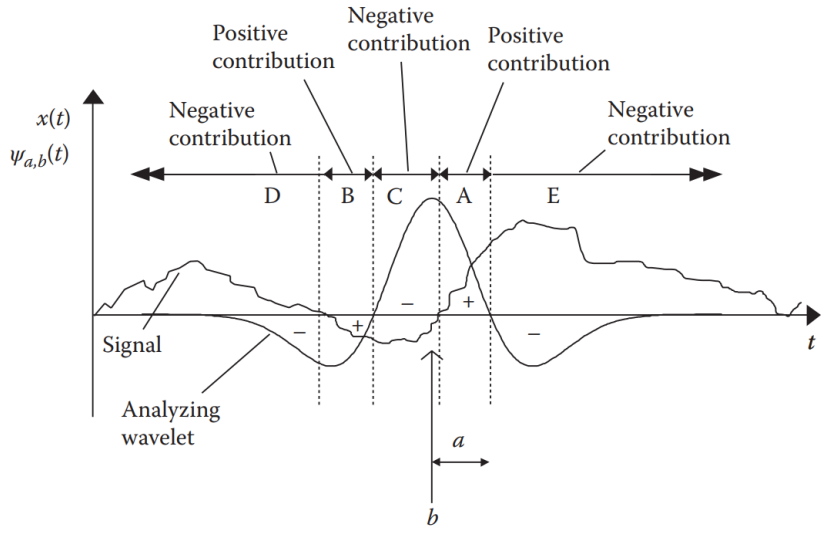


Figure 42 The wavelet of specific dilation and location on the signal. The regions which give positive and negative contributions to the integral are delineated in the sketch and marked with '+' and '-', respectively. Source: [2]

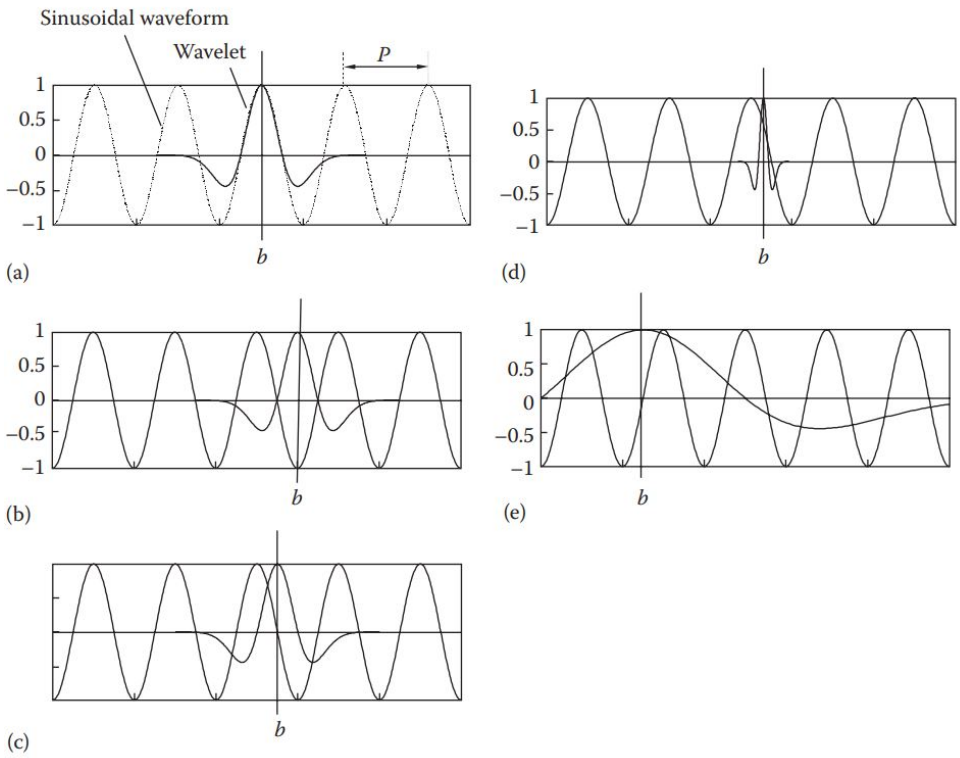


Figure 43 A wavelet interrogating a sinusoidal waveform: (a) The wavelet is in phase with the waveform giving good positive correlation. (b) The wavelet is now out of phase with the waveform giving good negative correlation. (c) The wavelet is again out of phase with the waveform, this time giving zero correlation. (d) The squeezed wavelet does not match the waveform locally. (e) The stretched wavelet does not match the waveform locally. Source: [2]

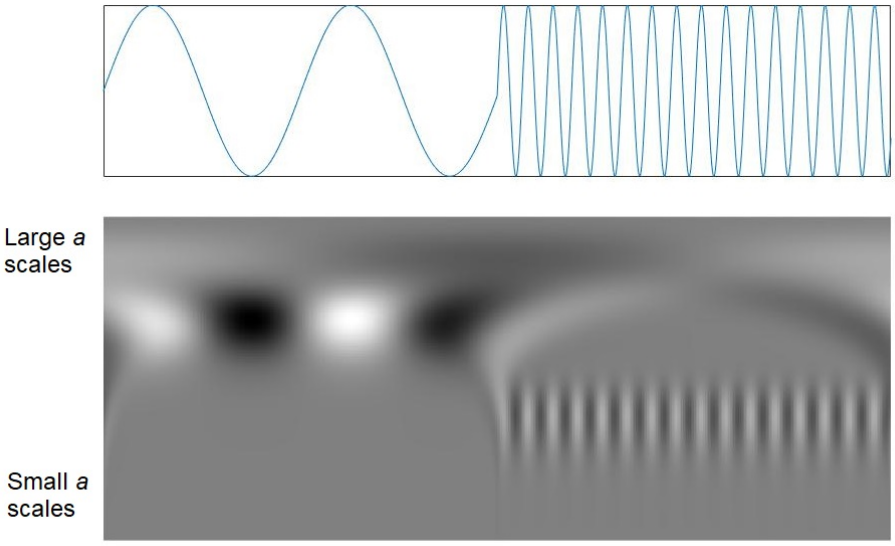


Figure 44 CWT plot of the signal above formed by two sines of different frequency concatenated. Source own.

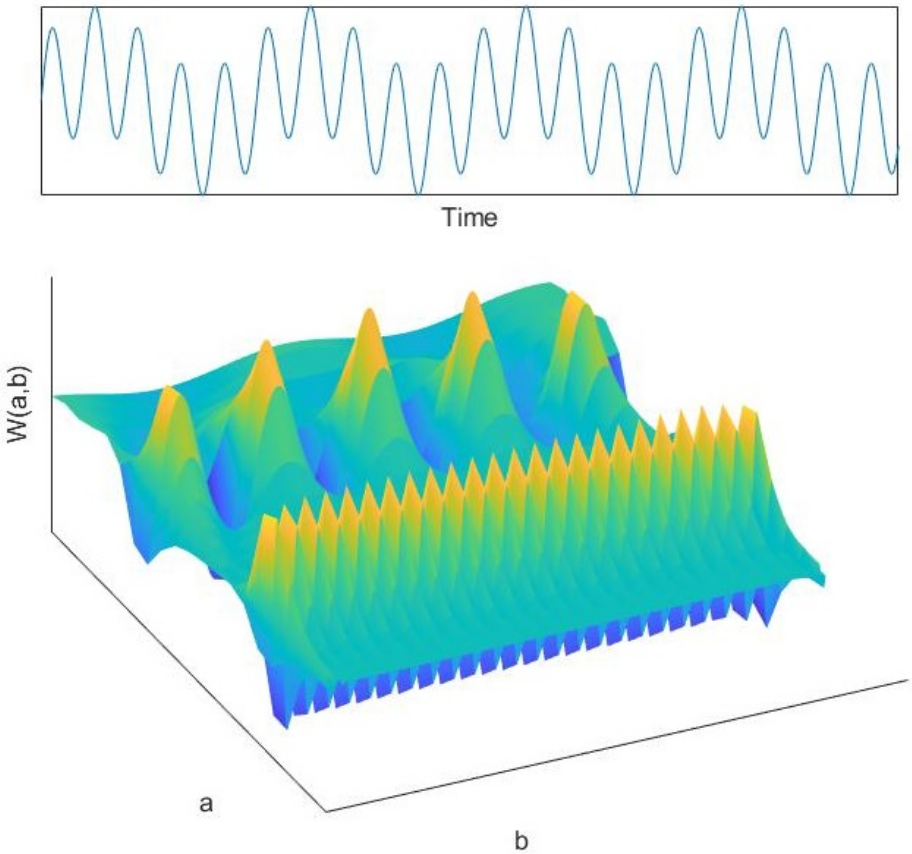


Figure 45 Continious wavelet transform plot of two combined sinusoids. Top: A signal composed of a combination of two sinusoids with different frequencies. Bottom: Surface plot of $\mathcal{W}(a, b)$. Source own.

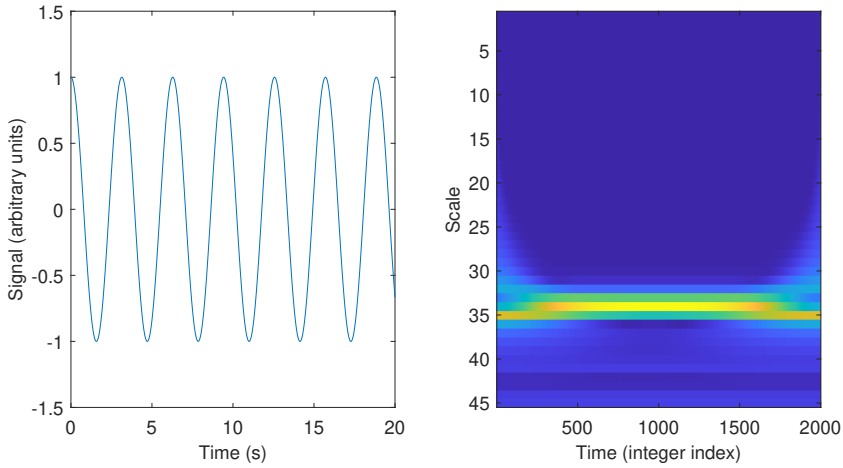


Figure 46 Continious wavelet transform plot of a single sinusoids. Left: Simple sinusoidal signal in time domain. Right: CWT plot of the signal . Source own.

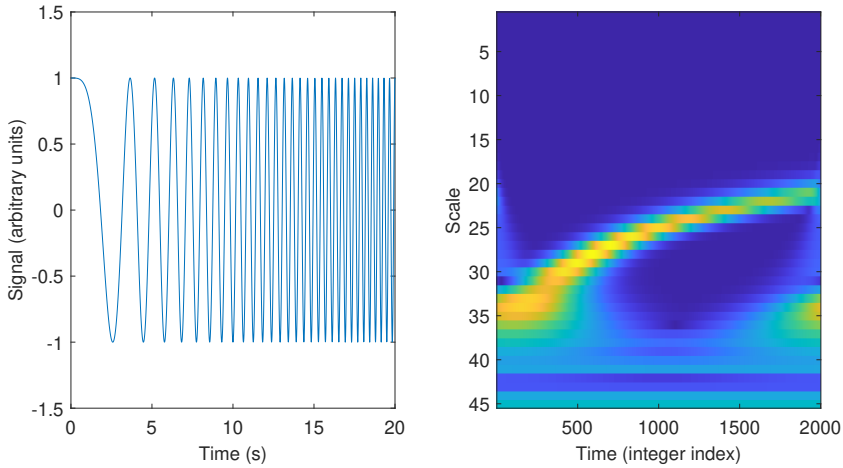


Figure 47 Continious wavelet transform plot of a linear chirp signal. Left: Linear swept-frequency cosine signal. Right: CWT plot of the signal. Source own.

8.2.1 The CWT in terms of the FT

The continuous wavelet transform can be expressed in terms of products of the Fourier transforms of the signal, $\hat{x}(f)$, and wavelet, $\hat{\psi}_{a,b}(f)$, using the convolution theorem.

$$\mathcal{W}(a,b) = \int_{-\infty}^{+\infty} \hat{x}(f)\hat{\psi}^*_{a,b}(f). \tag{49}$$

Expanding this equation can be written as follows. The full derivation can be found in [2] page 33.

$$\mathcal{W}(a,b) = \sqrt{a}\int_{-\infty}^{+\infty} \hat{x}(f)\hat{\psi}^*(af)e^{i(2\pi f)b}df. \tag{50}$$

This is an extremely effective breakthrough since it allows the CWT to be computed quickly using the fast Fourier transform (FFT) algorithm. Furthermore, the wavelet function's Fourier transform, $\hat{\psi}^*_{a,b}(f)$, is frequently known in analytical form, so it is not necessary to compute it using an FFT. Only the Fourier transform of the original signal needs to be calculated and it can be done with the methods previously seen in chapter 7. This way of performing CWT is schematically represented in Figure (48).

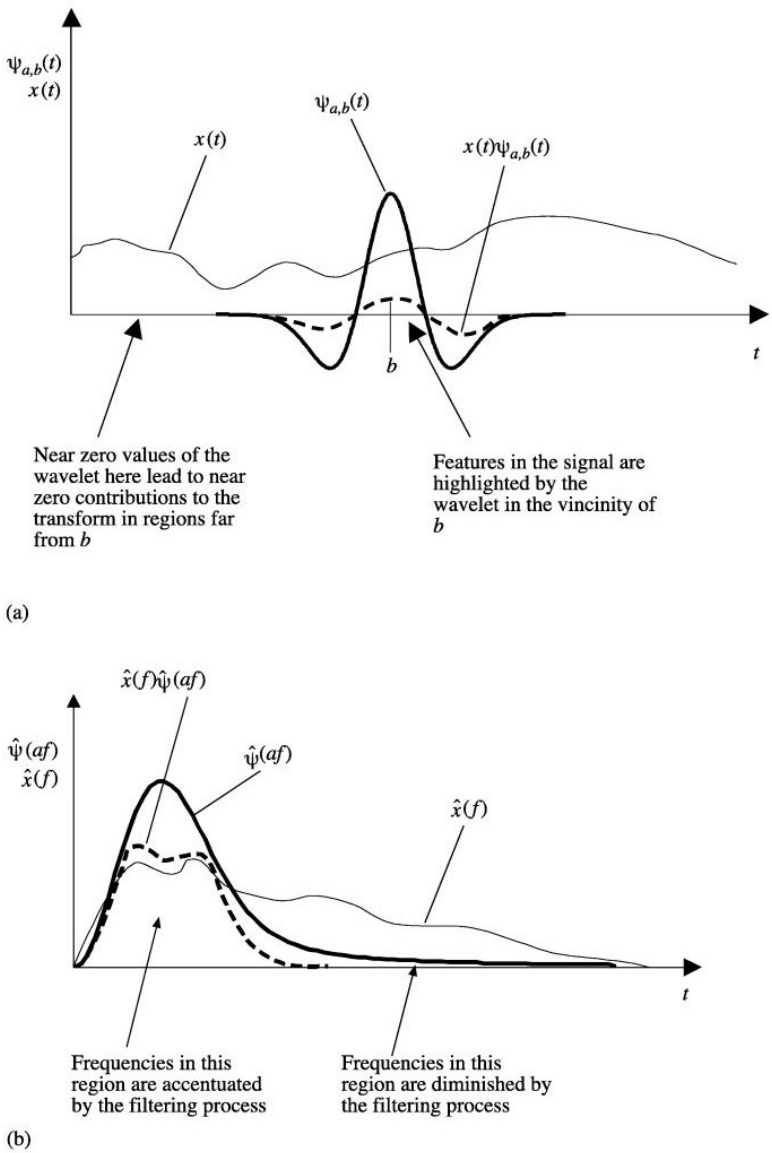


Figure 48 Schematic representation of the wavelet transform in its time and frequency representations. (a) The convolution of the wavelet with the signal. (b) The convolution in (a) expressed in the Fourier domain involves a product of the signal Fourier transform and the wavelet Fourier transform . Source: [2]

8.3 Discrete Wavelet Transform

The CWT, since the parameter a and b can be varied continuously, generates redundant information. The discrete wavelet transform reduces redundancy in wavelet coefficients across scales as much as possible while preserving the information in the original signal, resulting in a quicker wavelet transform and its inverse.

8.3.1 Discretization of Scale and Translation Parameters

Using a logarithmic discretization of the a scale and linking this to the magnitude of the steps taken between b locations is a natural technique to sample the parameters a and b. To connect b and a, we go to each b location in discrete steps that are proportional to the a scale. This type of discretization is expressed as

$$\begin{cases} a = a_0^m & a_0 < 1, b_0 < 0, m \in Z, n \in Z \\ b = nb_0a_0^m \end{cases} \tag{51}$$

where the wavelet dilation and translation are controlled by the number m and n respectively. a_0 is a fixed dilation step parameter and b_0 is the location parameter. The sign Z stands for an integer. The base wavelet's family is then stated as

$$\psi_{m.n}(t) = \frac{1}{\sqrt{a_0^m}}\psi\left(\frac{t - nb_0a_0^m}{a_0^m}\right). \tag{52}$$

A widely used wavelet discretization is the dyadic discretization which consists in generating a wavelet family set of orthonormal basis resulting in a more efficient discretization. The discrete wavelet parameters a_0 and b_0 are 2 and 1 respectively. [2]

The dyadic grid wavelet can be written as

$$\psi_{m.n}(t) = \frac{1}{\sqrt{2^m}}\psi\left(\frac{t - n2^m}{2^m}\right). \tag{53}$$

Each wavelet's product with all others in the same dyadic system is zero. This means that the information held in a wavelet coefficient is unique and can be used to regenerate the original signal completely without redundancy. In addition, the wavelets are normalized to have unit energy.

The discrete wavelet transform (DWT) can be written as

$$\mathcal{W}(m,n) = \int_{-\infty}^{\infty} x(t)\psi_{m,n}(t)dt. \tag{54}$$

And the inverse discrete wavelet transform (iDWT) as

$$x(t) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \mathcal{W}_{m,n}\psi_{m,n}(t). \tag{55}$$

8.3.2 Multiresolution Analysis

When Mallat was working on image processing in the 1980s, he came up with the notion of Multiresolution Analysis (MRA) [26]. The main idea of this type of compression is the following; given signal x(t), decomposition can be done to divide the signal into two sections (the detailed information in D1 and the approximate information in A1). The next level of detailed information in D2 and approximate information in A2 can be obtained by decomposing the approximate information in A1. This type of decomposition can be continued until the desired scale j is achieved.

First of all the scaling function needs to be defined.

$$\phi_{m,n}(t) = 2^{-m/2}\phi(2^{-m}t - n). \tag{56}$$

Unlike the wavelet function whose integral is zero, the integral of the scaling function is equal to one:

$$\int_{-\infty}^{\infty} \phi_{0,0}(t)dt = 1 \tag{57}$$

where $\phi_{0,0}(t)$ is known as the father wavelet or the father scaling function. A signal x(t) can be represented as follows using a mixed series expansion that includes both approximation and wavelet (detail) coefficients:

$$x(t) = \sum_{n=-\infty}^{\infty} S_{m_0,n}\phi_{m_0,n}(t) + \sum_{m=-\infty}^{m_0}\sum_{n=-\infty}^{\infty} \mathcal{W}(m,n)\psi_{m,n}(t) \tag{58}$$

where $\mathcal{W}(m,n)$ is the discrete wavelet tansform seen in equation (54) and $S_{m,n}$ are the approximation coefficients that can be calculated convolving the scaling function with the signal as follows:

$$S_{m,n} = \int_{-\infty}^{\infty} x(t)\phi_{m,n}(t)dt. \tag{59}$$

Equation (58) can then be rewritten as follows:

$$x(t) = x_{m_0}(t) + \sum_{m=-\infty}^{m_0} d_m(t) \tag{60}$$

where:

$$x_m(t) = \sum_{n=-\infty}^{\infty} S_{m,n}\phi_{m,n}(t), \tag{61}$$

$$d_m(t) = \sum_{n=-\infty}^{\infty} \mathcal{W}_{m,n}\psi_{m,n}(t). \tag{62}$$

From equation (60) the following equation can be easily deduced:

$$x_{m_1}(t) = x_m(t) + d_m(t). \tag{63}$$

This last equation is the heart of the multiresolution analysis. This equation expresses that if we add the signal detail at any scale (index m) to the approximation at that scale, we obtain the signal approximation at one level higher resolution.

8.3.3 Scaling Equation, Scaling Coefficients and Associated Wavelet Equation

The scaling equation (or dilation equation) expresses the scaling function $\phi(t)$ as contracted copies of itself and shifted by an integer step k along the time axis:

$$\phi(t) = \sum_k c_k \phi(2t - k) \tag{64}$$

where c_k is an associated scaling coefficient. We can prove that the scaling coefficients must fulfill the following requirement by integrating both sides of equation (64):

$$\sum_k c_k = 2. \tag{65}$$

Furthermore, we require that, in order to form an orthogonal system, the following conditions must be met:

$$\sum_k c_k c_{k+2k'} = \begin{cases} 2 & if \;\; k' = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{66}$$

To ensure that the wavelet and scaling functions are orthogonal, the same associated scaling coefficients are used to produce the associated wavelet equation as follows:

$$\psi(t) = \sum_k (-1)^k c_{1-k} \phi(2t - k). \tag{67}$$

Nonetheless, in this chapter, we will only discuss compact support wavelets which have a finite number of scaling coefficients, Nk. Modifying equation (67), the wavelet function for wavelets of compact support can be expressed as:

$$\psi(t) = \sum_k (-1)^k c_{N_k-1-k} \phi(2t - k). \tag{68}$$

Frequently, the wavelet function coefficients are rearranged in the following way for more compact writing.

$$b_k = (-1)^k c_{N_k-1-k}. \tag{69}$$

From equations (64) and (56), the scaling function at an arbitrary scale can be expressed as:

$$\phi_{m+1,n}(t) = \frac{1}{\sqrt{2}} \sum_k c_k \phi_{m,2n+k}(t). \tag{70}$$

Following the same procedure for the wavelet function the following equation can be obtained:

$$\psi_{m+1,n}(t) = \frac{1}{\sqrt{2}} \sum_k b_k \phi_{m,2n+k}(t). \tag{71}$$

These arbitrarily scaled scaling and wavelet functions are made up of a series of shifted scaling functions at the next smaller scale (one level higher). This structure follows the basic multiresolution analysis.

8.3.4 Haar Wavelet

To understand how the scaling and wavelet functions are truly obtained, the most simple example will be performed in this chapter, the Haar Wavelet. The scaling equation of the Haar Wavelet contains only two non-zero scaling coefficients. In other types of wavelets, the number of non-zero scaling coefficients varies. For example, in the Daubechies wavelet, the most commonly used orthogonal wavelets have between 2 and 20 non-zero scaling coefficients. Actually, the case of only 2 non-zero scaling coefficients (D2) is the Haar wavelet. Then, the scaling equation is given by:

$$\phi(t) = \phi(2t) + \phi(2t-1). \tag{72}$$

Solving equations (65) and (66) the scaling coefficients can be obtained. It is a system of two equations and two unknowns very easy to solve.

$$\begin{cases} c_0 + c_1 = 2 \\ c_0 c_0 + c_1 c_1 = 2. \end{cases} \tag{73}$$

Thus:

$$c_0 = c_1 = 1. \tag{74}$$

Returning to the scaling equation, its solution is a single block pulse.

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{75}$$

Following equation (68), the Haar wavelet equation can be obtained.

$$\psi(t) = \phi(2t) - \phi(2t-1). \tag{76}$$

Finally, the Haar wavelet can be obtained and the result is the two-unit block pulses, one positive and one negative, seen in previous sections.

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{77}$$

Figure (49) *a* and *b* shows the building process of the scaling and wavelet function as well as the differences between them. The scaling and wavelet functions are the new bases on which the

information is represented. This can be related to the DCT bases functions from section 7.5. The Haar scaling and wavelet function can be compared to the DCT basis functions of a 2-by-2 matrix in Figure (19). The scaling function expresses the mean of the signal, defined in the DCT section as the DC coefficient (B_{00}) while the wavelet function is the other coefficients depending on the direction of propagation of the wave. In the 2-by-2 matrix case in Figure (19), the scaling function would take the role of the top left base. On the other hand, the wavelet function is the top right base for propagation direction on the x-axis, the bottom left for the y axis, and the bottom right for both directions.

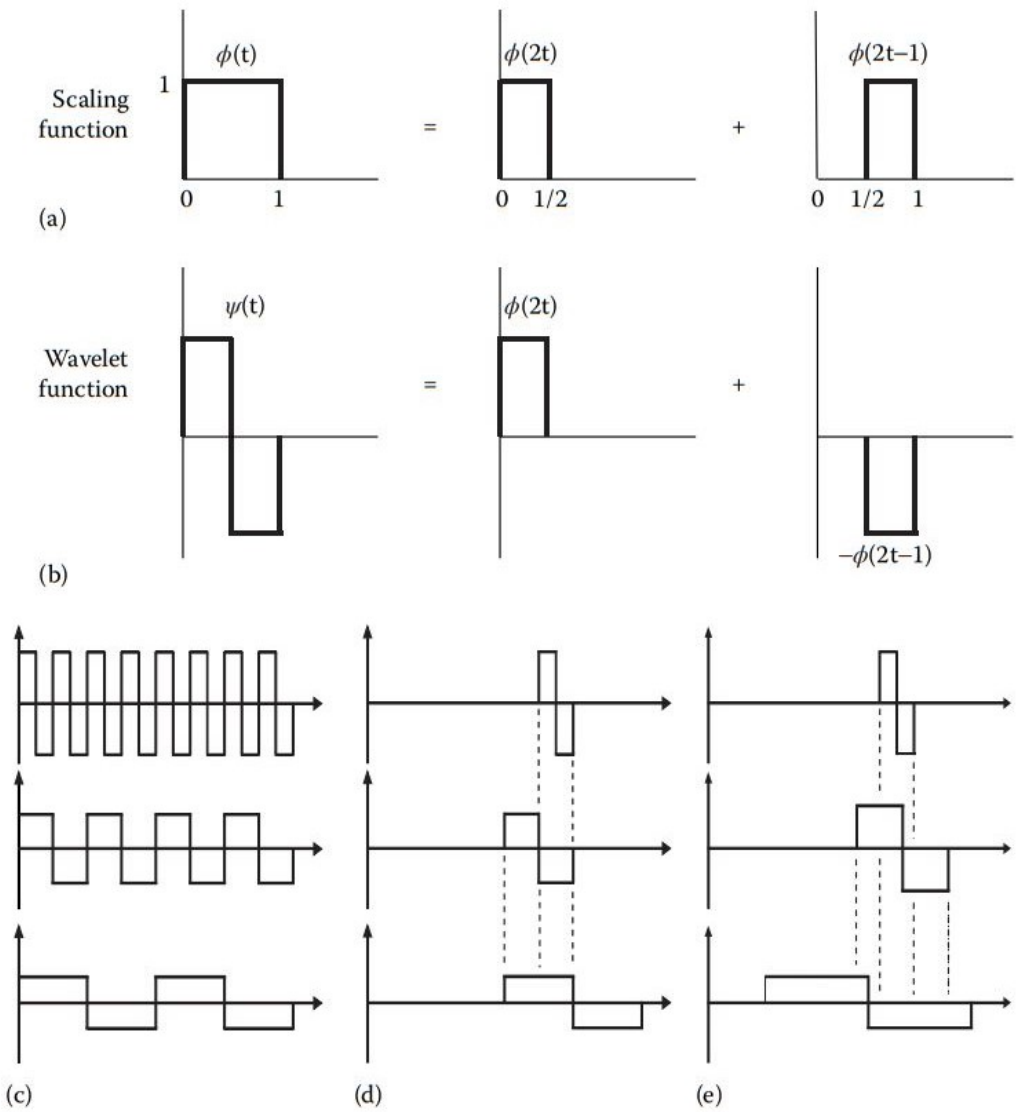


Figure 49 Discrete orthonormal wavelets: (a) The Haar scaling function in terms of shifted and dilated versions of itself. (b) The Haar wavelet in terms of shifted and dilated versions of its scaling function. (c) Three consecutive scales shown for the Haar wavelet family specified on a dyadic grid, for example, from top to bottom: $\psi_{m,n}(t)$,$\psi_{m+1,n}(t)$,$\psi_{m+2,n}(t)$. (d) Three Haar wavelets at three consecutive scales on a dyadic grid. (e) Three Haar wavelets at different scales. This time, the Haar wavelets are not defined on a dyadic grid and are hence not orthogonal to each other. Source: [2]

8.3.5 Daubechies wavelets

As with any other orthogonal wavelet, the scaling functions associated with these wavelets must meet the requirements indicated by the equations (65) and (66). In addition to these requirements, Daubechies wavelets demand to have compact support and be smooth to some extent. The wavelet's smoothness is determined by a moment condition, which is defined in terms of scaling coefficients as

$$\sum_{k}^{N_k-1} (-1)^k c_k k^m = 0 \qquad m = 0, 1, ..., N_k/2 - 1 \tag{78}$$

where N_k is the number of scaling coefficients. These wavelets have $N_k/2$ vanishing moments, meaning they can suppress polynomial sections of the signal up to degree $N_k/2-1$. This indicates that Daubechies wavelets are excellent at capturing polynomial behavior in signals. Next we will obtain the scaling coefficients for the Daubechies wavelets for 4 non-zero scaling coeficients, also known as D4. From equation (64), the scaling equation for a wavelet with four coefficients is:

$$\phi(t) = c_0\phi(2t) + c_1\phi(2t-1) + c_2\phi(2t-2) + c_3\phi(2t-3) \tag{79}$$

and, according to equation (67) , the equivalent wavelet function is:

$$\psi(t) = c_3\phi(2t) - c_2\phi(2t-1) + c_1\phi(2t-2) - c_0\phi(2t-3). \tag{80}$$

As in section 8.3.4 for the Haar Wavelet, from equations (65) and (66) the following equations can be obtained to define the scaling coefficients.

$$\begin{cases} c_0 + c_1 + c_2 + c_3 = 2 \\ c_0^2 + c_1^2 + c_2^2 + c_3^2 = 2. \end{cases} \tag{81}$$

Right now we have two equations and four unknowns. From the moment condition of equation (78), the following two equations can be found defining the system as a system with a single unique solution.

$$\begin{cases} c_0 - c_1 + c_2 - c_3 = 0 & \text{for } m = 0 \\ -1c_1 + 2c_2 - 3c_3 = 0 & \text{for } m = 1. \end{cases} \tag{82}$$

Four scaling coefficients which satisfy equations (81) and (82) are:

$$c_0 = \frac{1+\sqrt{3}}{4} \qquad c_1 = \frac{3+\sqrt{3}}{3} \qquad c_2 = \frac{3-\sqrt{3}}{4} \qquad c_3 = \frac{1-\sqrt{3}}{3}$$

and also

$$c_0 = \frac{1-\sqrt{3}}{4} \qquad c_1 = \frac{3-\sqrt{3}}{3} \qquad c_2 = \frac{3+\sqrt{3}}{4} \qquad c_3 = \frac{1+\sqrt{3}}{3}$$

8.3.6 Mallart Algorithm (wavelet packet decomposition)

This algorithm effectuates a multiresolution decomposition of the input signal x(t) [27]. The wavelet and scaling functions aren't used in this algorithm; instead, the filters h and g that characterize their interaction are used. The filters h(n) and g(n) are called a pair of low-pass and high-pass wavelet filters respectively and can be defined as follows:

$$h(n) = \langle \phi, \phi_{-1,n} \rangle \tag{83}$$

$$g(n) = \langle \psi, \phi_{-1,n} \rangle. \tag{84}$$

The entire development of this algorithm can be consulted in the book *The Illustrated Wavelet Transform Handbook* [2].The main idea is to decompose the signal x(t) into low and high-frequency components as follows:

$$a_{j,k} = \sum_m h(m-2k) a_{j-1,m} \tag{85}$$

$$d_{j,k} = \sum_m g(m-2k) a_{j-1,m}. \tag{86}$$

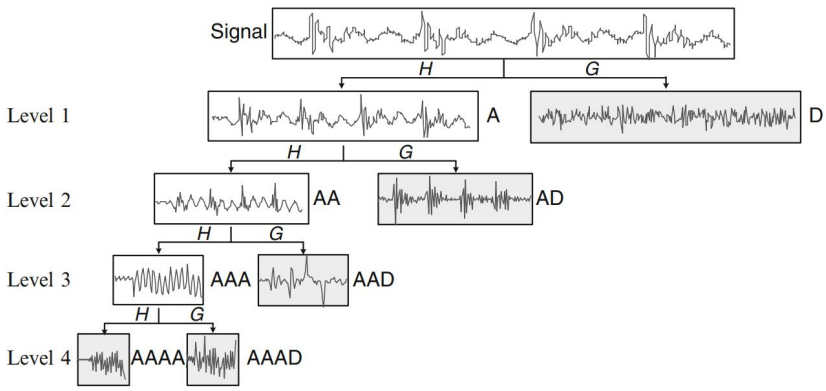


Figure 50 Procedure of a four-level signal decomposition using discrete wavelet transform. Note: H low-pass filter, G high-pass filter, A approximate information, D detailed information . Source: [2]

8.4 Second-generation wavelet transform: Lifting scheme

The CWT and DWT are referred as first-generation wavelet transforms. This type of transformation works well for infinite or periodic signals, but it's unclear how to adapt it for constrained domain applications. To solve this issue Wim Sweldens introduced in 1996 the lifting scheme known as a second wavelet generation [36]. The lifting scheme does not use any Fourier analysis and the wavelet is no longer translated and dilated. This algorithm is fast in terms of computation and is advantageous for reverse transformation. [7]

The lifting scheme consists mostly of three key operational steps: (1) splitting, (2) prediction, and (3) updating. [36, 18, 10]

Splitting: The original signal is divided into two subsets that do not intersect. The split effect is improved by a high correlation between the subsets. Digital signals are highly correlated, and the correlation structure is local [9]. This means that the function slowly varies so close points are closely related. Therefore, signal sequences are usually divided into odd and even sequences as follows:

$$signal = \begin{cases} x_{odd} & = x(2i-1) \\ x_{even} & = x(2i) \end{cases}, \quad i = 1, 2, 3, .., n.$$

Prediction: Using the similarity of data, we can predict the x_{odd}, from the neighboring even coefficients,x_{even}, by using a predict operator P, which is independent of the dataset. Each predictor is a linear combination of adjacent even coefficients.

$$P(x_{even})[n] = \sum_l p_l x_{even}[n+l]. \tag{87}$$

The difference between the predicted result and the odd sample is referred to as the detail of the original signal, d. If the prediction is accurate, the prediction residuals will be small and will contain the same information as the original dataset. This prediction step is equivalent to applying a high-pass filter to the original signal. The predict procedure can be defined as follows:

$$d[n] = x_{odd}[n] - P(x_{even})[n]. \tag{88}$$

Updating: By using the update operator U, which is also independent of the dataset, the even samples are updated for the coarse approximation to have the same average as the original signal. This updating step is equivalent to applying a low-pass filter to the original signal. The operator U is a linear combination of neighboring d values.

$$U(d)[n] = \sum_l u_l d[n+l]. \tag{89}$$

This update's result is referred to as the approximation, a, and it can be calculated as follows:

$$a = x_{even} + U(d). \tag{90}$$

The signal is divided into two pieces using the forward process mentioned above: approximation and detail. The decomposition can be repeated iteratively by using the approximation as the input signal, where each iteration is described as a level. Wavelet packet transform can also be performed using the lifting method by combining iterated decomposition of the detail and approximation.

No information is lost because each lifting step is always invertible. By using inverse second generation wavelet transform, the original data can be perfectly recovered (ISGWT). Both the prediction operator and the updating operator are used in the reverse method, just as they are in the forward procedure. The original signal can be recovered as follows:

$$\begin{cases} x_{odd} & = d + P(x_{even}) \\ x_{even} & = a - U(d). \end{cases}$$

By combining x_{even} and x_{odd}, the signal may be reconstructed. Figures (51) and (52) show diagrams of the forward and reverse transform procedures of the lifting scheme, respectively. Figure (53) explains graphically the forward transform.

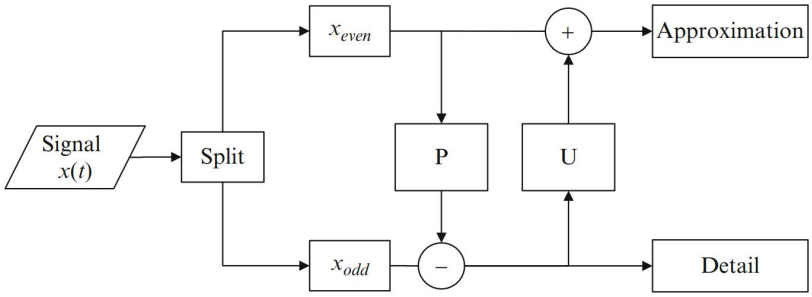


Figure 51 Forward transform procedure of lifting scheme. Source: [18]

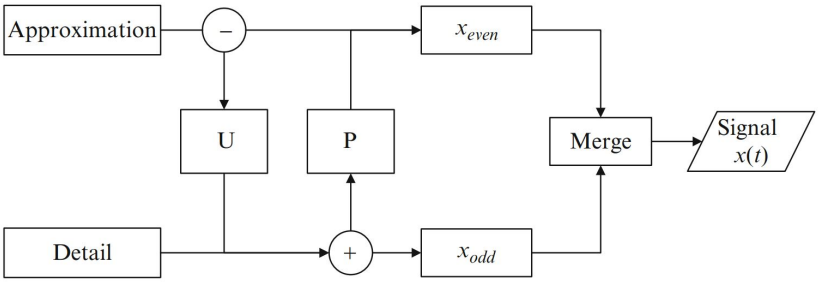


Figure 52 Reverse transform procedure of lifting scheme. Source: [18]

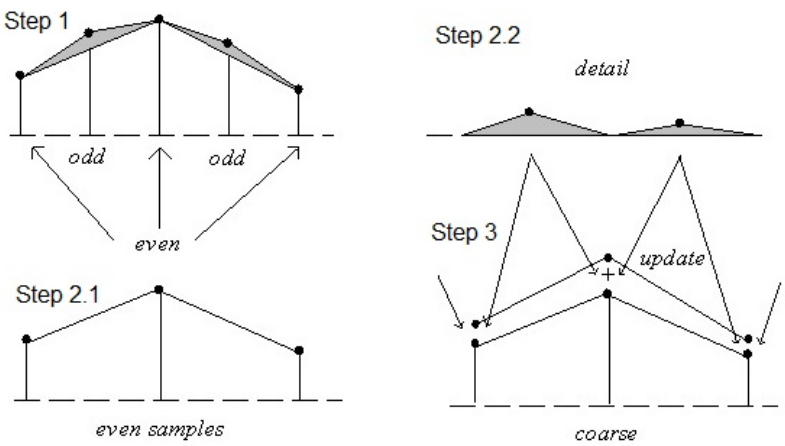


Figure 53 An illustration of the forward transform. Step 1: the signal is split into even and odd parts. Step 2.1 and 2.2: the odd part is predicted linearly by the even part; the detail is what is missed by the prediction. Step 3: the calculated detail is used to update the even signal in order for the coarse signal to have the same average as the original signal. Source: [31]

9 Principal component analysis (PCA)

Principal component analysis, PCA, is the simplest and most widely used technique for dimensionality reduction. The basic idea is to obtain a linear and orthogonal projection of high-dimensional data to a low-dimensional subspace that is a fair approximation to the original data (see Figure (54)). The PCA constructs a linear transformation that chooses a new coordinate system for the original data set in which the largest variance of the data set is captured on the first axis, the first major component. The second-largest variance is the second axis, second major component, and so on [28]. Figure (55) illustrate this transformation.

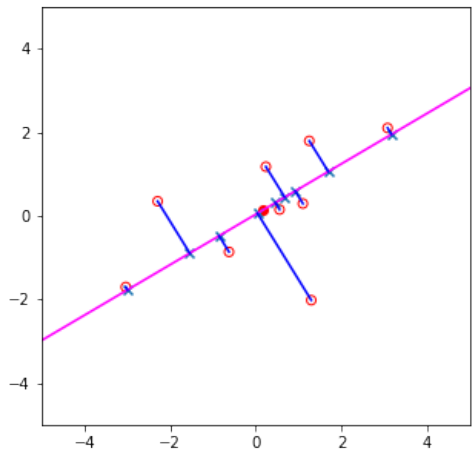


Figure 54 An illustration of PCA where we project from 2D to 1D. Circles are the original data points, crosses are the reconstructions. The red star is the data mean. Code to generate the figure. Source: [28]

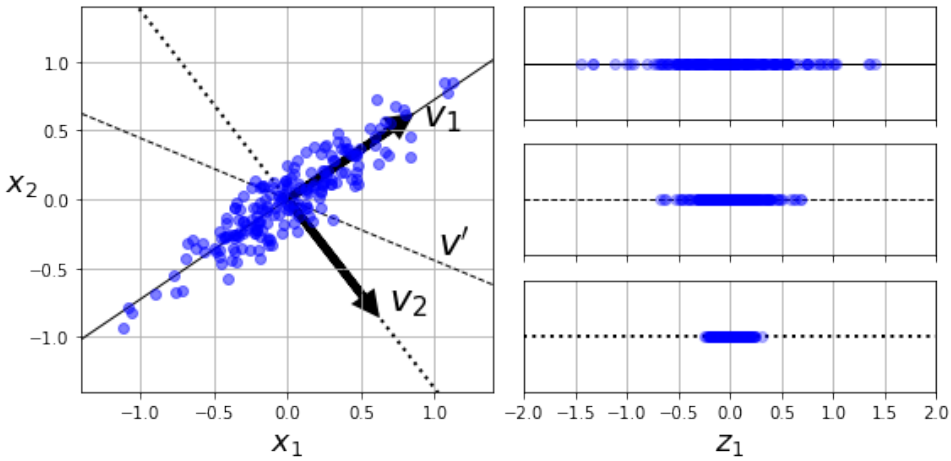


Figure 55 Illustration of the variance of the points projected onto different 1D vectors. v_1 is the first principal component, which maximizes the variance of the projection. v_2 is the second principal component which is direction orthogonal to v_1. Finally v' is some other vector in between v_1 and v_2. Code to generate the figure. Source: [28]

9.1 Singular value decomposition (SVD)

The goal of the PCA is to find a matrix T that is an approximation of X created with L components on $1 <= L <= m$. The smallest possible Frobenius norm is desired.

$$min_T \ \|X - T\|^2. \tag{91}$$

The PCA can be associated with another type of factorization of matrices, the singular value decomposition (SVD). The singular value decomposition consists of factorizing an X matrix of dimensions $n \times m$ as follows:

$$X = U\Sigma V^T \tag{92}$$

where U is a unitary matrix of dimensions $m \times m$, Σ is a diagonal matrix of dimensions $m \times n$ and V is a unitary matrix of dimensions $n \times n$. The main diagonal entries of Σ are known as the singular values of X, σ_r. The m columns of U and the n columns of V are called left singular vectors and right singular vectors of X, respectively.

The matrix $X^T X$ can be expressed as:

$$X^T X = V\Sigma^T U^T U\Sigma V^T,$$
$$X^T X = V\Sigma^T \Sigma V^T,$$
$$X^T X = V\hat{\Sigma}^2 V^T.$$

Using the SVD the matrix T can be written as:

$$T = XV,$$
$$T = U\Sigma V^T V,$$
$$T = U\Sigma.$$

Therefore, each column of T is given by one of the left singular vectors of X multiplied by the corresponding singular value. Similar to the eigenvalue decomposition, a truncated matrix $n \times L$ can be obtained by considering only the first L largest singular values and their singular vectors:

$$T_L = U_L \Sigma_L = X_L V_L.$$

The truncation of an X or T matrix that uses a decomposition into singular values truncated in this way produces a truncated matrix that is the closest possible matrix of rank L to the original matrix, in the sense of the difference between the two that they have the smallest possible Frobenius norm.

$$X \approx U_L \Sigma_L V_L^T. \tag{93}$$

To fully understand how SVD works and how to use it for image compression some examples will be done following the directions in the book *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* by Steven L. Brunton and J. Nathan Kutz.

Example 1: Noisy Gaussian Data

For the first example we will create or own data. By selecting 10,000 vectors from a two-dimensional normal distribution with zero mean and unit variance, a noisy cloud of data points is created. These vectors are then rotated by $\pi/4$ and scaled in the x and y directions by the values in Table 2. Finally, the entire data cloud is translated to $x_C = [1, 2]^T$, giving it a nonzero center. See Figure (56).

The singular values obtained from the SVD are shown in Table 2. These values match the data scaling values that had been used to create the data. The matrix U from the SVD is also very similar to the rotation matrix $R_{\pi/4}$ used to rotate the initial data. As can be seen, the SVD is able to obtain the rotation of the data and the directions in which the variance of the projection is maximized.

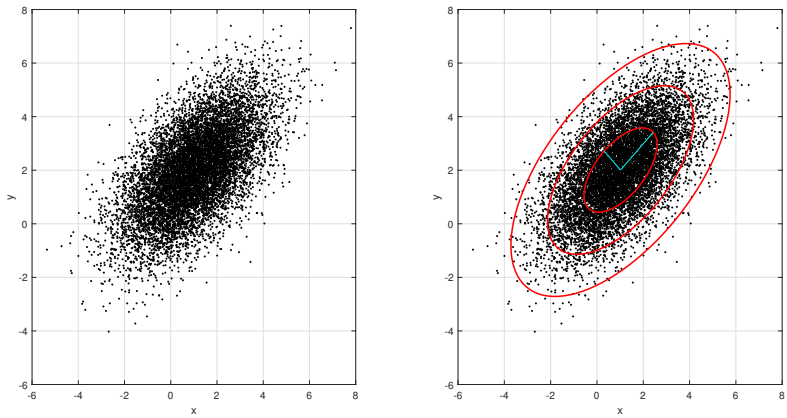


Figure 56 Principal components capture the variance of mean-subtracted Gaussian data shown in the left. The first three standard deviation ellipsoids (red), and the two left singular vectors, scaled by singular values shown in cyan. Image generate by code from: [5]

$$R_{\pi/4} \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix} \qquad U = \begin{bmatrix} 0.7086 & -0.7056 \\ 0.7056 & 0.7086 \end{bmatrix}$$

Table 2 Standard deviation of data and normalized singular values.

	σ_1	σ_2
Data	2	1
SVD	1.992	0.998

Example 2: Ovarian Cancer Data

In this second example, we will use the MATLAB database *'ovariancancer'* from the *Statistics and Machine Learning Toolbox*. This database contains the observations of 216 subjects and 4000 predictors. Unlike in the previous example, we do not know the distribution of the data and we can not easily represent it. In order to visualize in a three-dimensional space whether

or not a person has ovarian cancer, we will do the PCA and graph the data with the first three main components. As we can see in Figure (57), with only three predictors we can make a pretty good separation between the two groups.

To confirm that we are indeed using the most significant components we can plot the value of the variances for each component. To confirm that the error we are assuming is small we can plot the accumulated energy with each component. As we can see in Figure (58), indeed the first components are the most significant, and as the r increases, their weight decreases significantly. In the energy graph we can see that with three components the accumulated energy is 0.62.

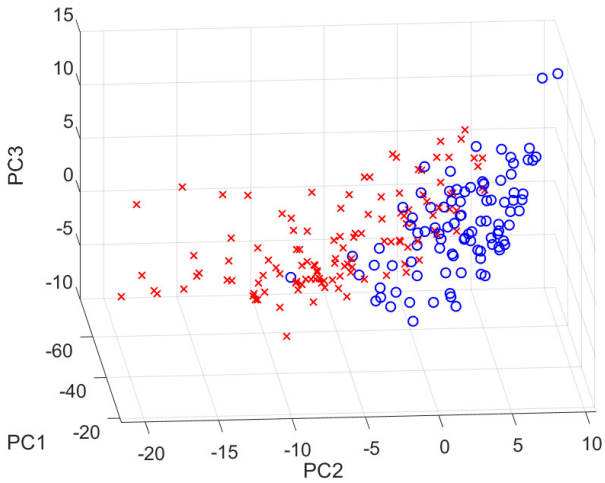


Figure 57 Distribution of points in the three main components. The red x's are the people with cancer and the blue circles are those who do not. Image generate by code from: [5]

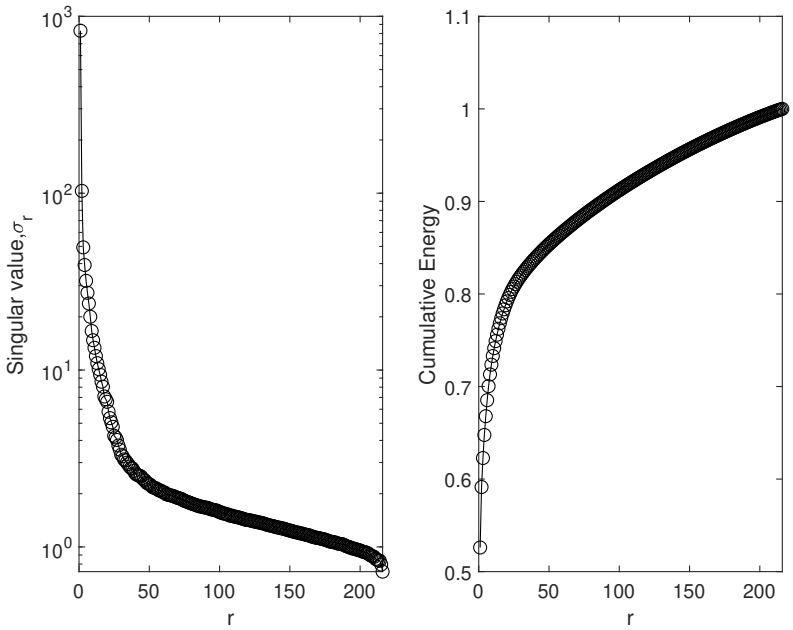


Figure 58 Singular values and cumulative energy for the Ovarian cancer data.

9.2 Image compression with PCA

In the study of an image we can do a procedure quite similar to what we did in the previous sections. A two-dimensional X array with the value of the intensity of each pixel will be entered into the SVD function of MATLAB. This function will return the matrices U, Σ and V. Truncating these matrices to a desired number r, we can obtain an approximate X matrix according to equation (93).

In Figure (59) we can see the original 1598×1600 image with 10, 100 and 500 components. With only 10 components we can recognize the silhouette of the cat, but the image is not well defined. With 100 components the image seems almost perfectly reconstructed, but the details of the image, like the whiskers, can be improved. With 500 components the image is almost identical to the original. In Figure (60) we can see the variation σ_r and the accumulated energy for each r. In this case the first components are also of a much higher order than the others and with less than 500 components the accumulated energy is approximately 0.95.

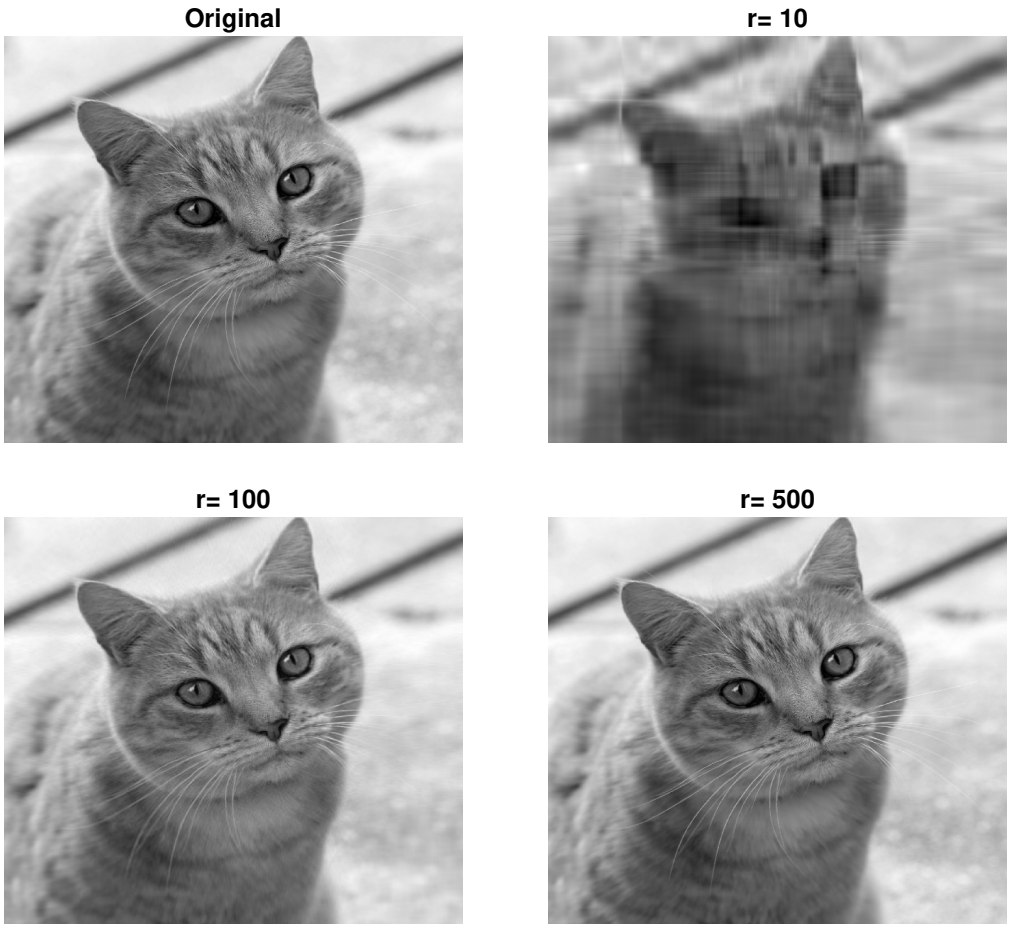


Figure 59 Original image and the approximated with r elements.

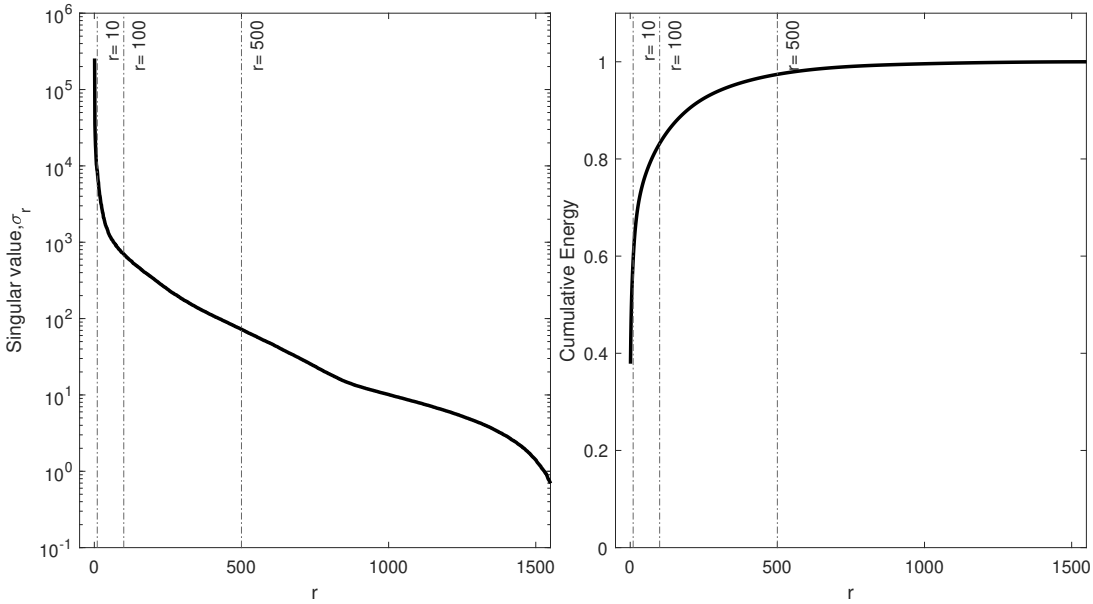


Figure 60 Singular values and cumulative energy for the image from figure 59.

9.3 Video compression with PCA

All the videos used in this sections can be visualized and downloaded here: Link videos.

Video compression can be easily done with the PCA technique. We will approach the problem in a similar way as in the case of images, but in this case we will reorder the pixels of each frame in a single vector. Next we can order these vectors in two ways, by rows or by columns. The PCA is designed in such a way that it reduces the dimensionality of the features. So we will be interested in ordering our matrix in such a way that we have more columns than rows.

The video of study can be downloaded in the link at the top of this page and it is named *'VideoOriginal'*. It is a 10 second video at 25 fps (250 frames) with a resolution of 360×480 (172800 pixels), therefore we want the pixels to be the columns and the frames to be the rows. This will be our case 1.

Case 1: Pixels are columns and frames are rows.

Three different approximations have been carried out for different compression levels (r equal to 10, 50 and 100). These videos are inside the folder *VideoCompressionCase1*. With 10 features we can only vaguely recognize the silhouette of the person and the plane, but the resolution is terrible. With 50 features, the video is already recognizable, but a series of undulations appear throughout the image, above all they can be seen near the plane. It can be an acceptable compression for depending on what uses. At 100 features these ripples are still present but much less visible. In Figure (61) we can see the variation σ_r and the accumulated energy for each r.

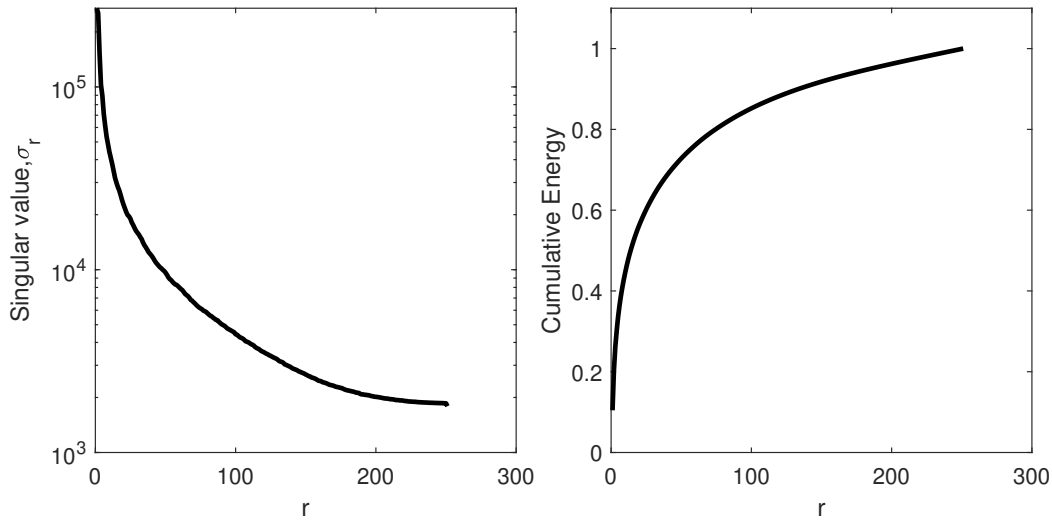


Figure 61 Singular values and cumulative energy for the video named *'VideoOriginal'*.

Case 2: Frames are columns and pixels are rows.

Next we will introduce the transpose of the input matrix of case 1. As said before, this case will have worse compression performance, but it will allow us to visualize what we will call the 'eigenframes'. Now each column of X represents a whole frame. In this way each column of the matrix U also represents a frame, an eigenframe. The most common frame correlations are represented by these images, see Figure (62). This decomposition produces a set of eigenframes that define a new coordinate system. The dot product of each of the main components can be used to represent images in these coordinates.

As in the other case, three different approximations have been carried out for different compression levels (r equal to 10, 50 and 100). These videos are inside the folder *VideoCompressionCase2*. These videos all look worse than their respective ones in Case 1. This agrees with the Figure (63) where the the Case 2 singular value and cumulative energy is under the Case 1 values.

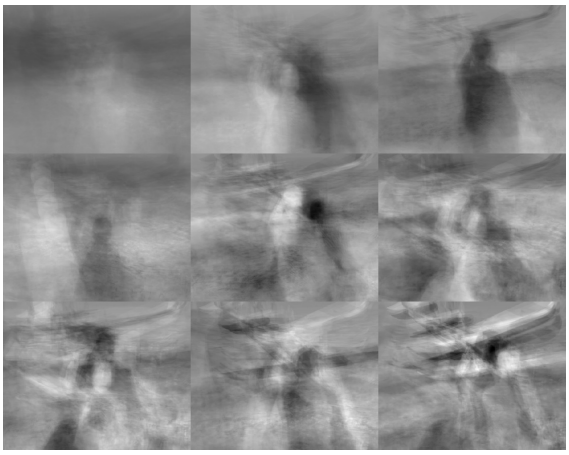


Figure 62 The first 8 main components (eighenframes) for the video named *'VideoOriginal'*

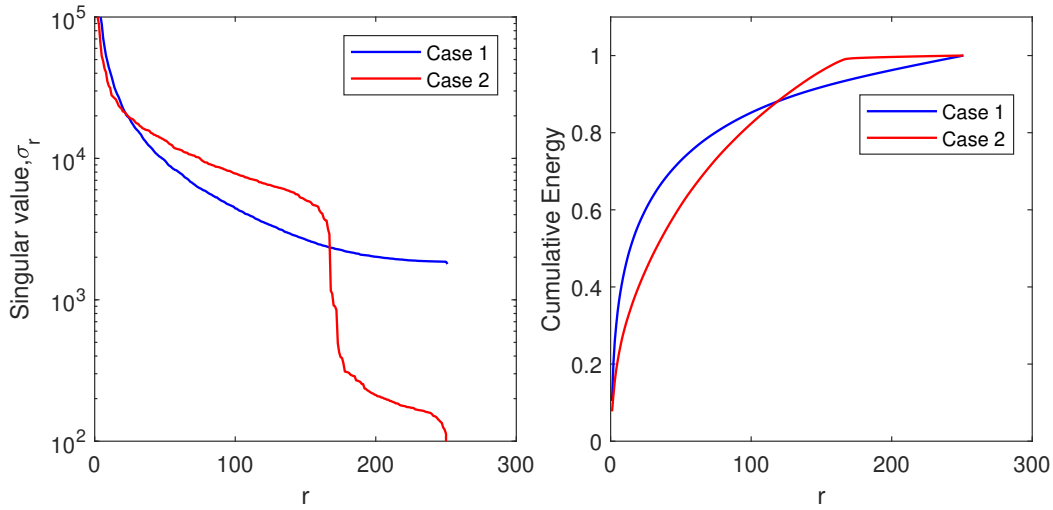


Figure 63 Singular values and cumulative energy for the video named *'VideoOriginal'* for Case 1 and 2.

9.4 EigenFaces

Following in the direction of case 2 of section 9.3, the same procedure can be done but instead of frames, we can use a database of images of different faces. In this section, a database of 15747 images of cat faces with a resolution of 64×64 has been used. Some examples of these images can be seen in Figure (64).

As in the case 2 of section 9.3, each column of the matrix U represents an image of a face, an eighenface. The eigenfaces themselves form a basis set of all images and by allowing the smaller set of basis images to represent the original training images, dimension reduction is achieved. The most important bases can be visualized in Figure (65). The new bases created by PCA can be used for facial recognition [33, 37]. In Figure (66) we can see the variation σ_r and the accumulated energy for each r.

Figure 64 Some examples of the images used in this section. (This database can be downloaded here: Cat faces database)

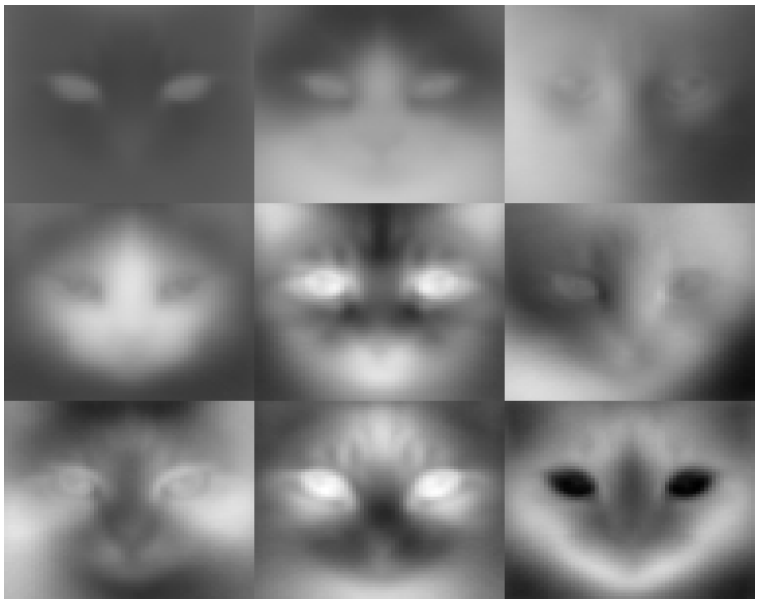


Figure 65 The first 8 main components (eighenfaces) of the database used in this section

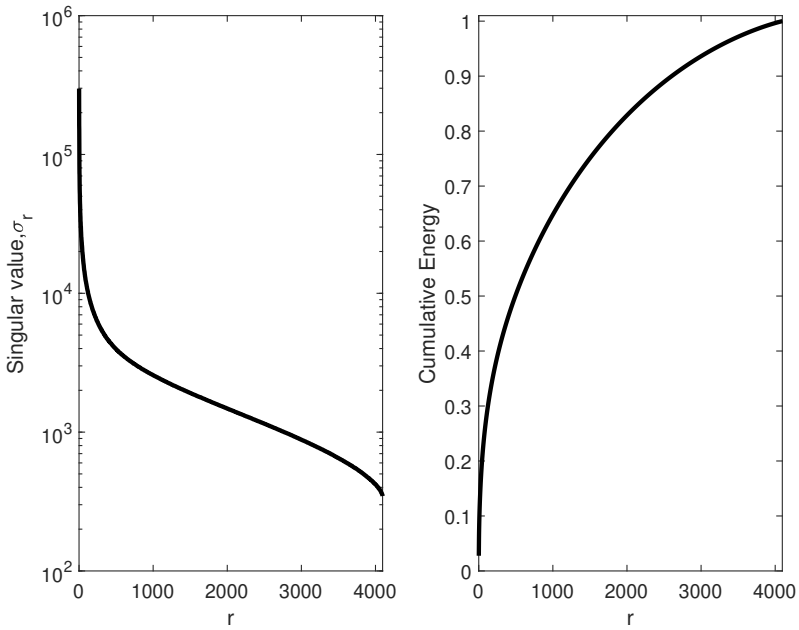


Figure 66 Singular values and cumulative energy for the database used in this section.

10 Comparison between methods

Next, we will compare the compression efficiency of the three methods studied throughout this project (Fourier, Wavelets, and PCA). Various ways of performing both Fourier and Wavelet compression have been explained, but not all of them will be compared. Only the most important and useful for image compression will be studied.

Normally, the amount of compression is expressed as a ratio but in this paper it will be expressed as a percentage of the stored data compared to the original image data.

10.1 Differences between images

To study image compression we must be able to compare two images and be able to quantify how similar they are. The best method of quantifying visual image quality for applications where images are ultimately to be viewed by humans is through subjective evaluation. But our goal is to find a quantitative measure that can predict subjective image quality automatically.

The easy and simple way would be to use the mean absolute error (MAE) or the mean squared error (MSE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i| \,, \tag{94}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - x_i)^2. \tag{95}$$

However, two distorted pictures with the same MAE or MSE might have a huge variety of faults, some of which are more noticeable than others. To solve this problem, the Structural Similarity Index (SSIM) can be used. Natural image signals are highly structured, with significant dependencies between pixels, especially when they are spatially close together, and these relationships contain crucial information about the organization of the objects in the visual scene. The goal of our new method is to compare the structures of the reference and reconstructed signals. The SSIM formula is based on three comparison measurements between the samples of x and y: luminance (l), contrast (c), and structure (s). The three components are added together to provide an overall similarity score which ranges between 1 and 0. The more similar the two images are, the closer the SSIM score is to 1. The three components are generally unrelated; for example, changes in structures and brightness will have little impact on picture contrast.

The individual comparison functions are:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{96}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{97}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x + \sigma_y + c_3} \tag{98}$$

where:

- μ_a is the average of a
- σ_a^2 is the variance of a
- σ_{ab} is the covariance of a and b
- $c1 = (k_1 L)^2$, $c2 = (k_2 L)^2$, $c_3 = c_2/2$ are two variables to stabilize the division with weak denominator
- L is the dynamic range of the pixel-values
- $k1 = 0.01$ and $k2 = 0.03$ by default

SSIM is then a weighted combination of those comparative measures:

$$SSIM(x,y) = [l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma], \tag{99}$$

Setting the weights α ,β ,γ to 1, Equation (99) can be reduced to

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c2)}. \tag{100}$$

When evaluating image quality, it's better to utilize the SSIM index locally rather than generally for several reasons. First of all, at normal viewing distances, the human observer can only comprehend a small portion of the image with high resolution at any given moment. By imitating the way we humans interpret images we can obtain more accurate results with our perception of the visual field. Secondly, image statistical characteristics are typically non-stationary in space. Lastly, image distortions might be spatially variable. The mean SSIM (MSSIM) index can be used to evaluate the overall image quality:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j) \tag{101}$$

where X and Y represent the reference and reconstructed images, respectively; xj and yj represent the image contents in the j-th local window, and M is the number of samples in the quality map. Figure (67) shows a comparison between the MSE and the MSSIM performance. The results of the MSE and MSSIM measurements are listed in the figure caption. Clearly, MSE fails miserably in this scenario. The MSSIM results are substantially more consistent with the visual look of the qualitative data.

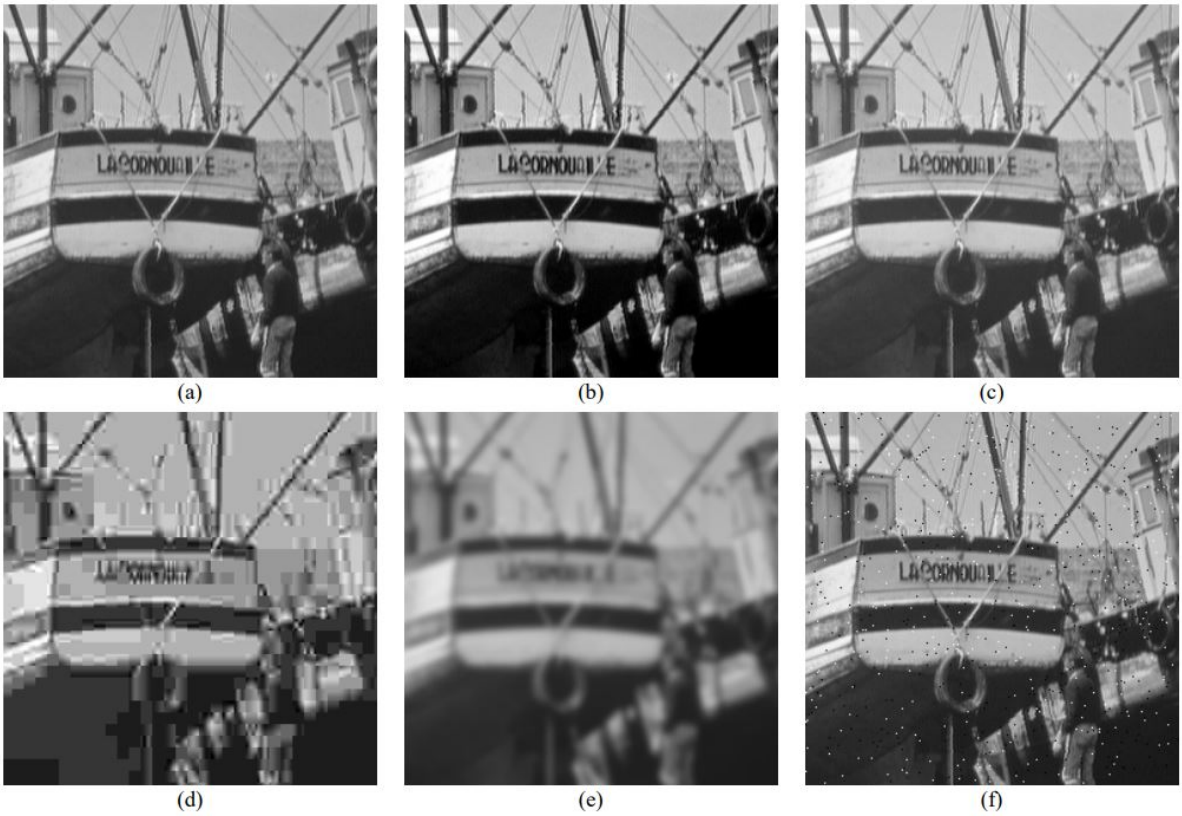


Figure 67 Comparison of "Boat" images with different types of distortions, all with MSE = 210. (a) Original image (8bits/pixel; cropped from 512×512 to 256×256 for visibility); (b) Contrast stretched image, MSSIM = 0.9168; (c) Mean-shifted image, MSSIM = 0.9900; (d) JPEG compressed image, MSSIM = 0.6949; (e) Blurred image, MSSIM = 0.7052; (f) Salt-pepper impulsive noise contaminated image, MSSIM = 0.7748. Source: [38]

10.2 Comparison between Fourier transform methods

In this section, the performance of three different Fourier transform methods will be studied. The chosen methods are FFT, DCT, and JPEG because they have the greatest importance in the field of image compression. The same image will be compressed via threshold at different compression rates for each method. Keep in mind that the FFT results in complex numbers where both the real and imaginary parts are important to reconstruct the original image. Although mathematically a complex number is a single number, computers need two numbers to represent them (one for the real part and one for the imaginary part). This detail has been taken into account when making the compression. The two images used for this test can be seen in Figure (68).

The results of this comparison can be seen in Figures (69) and (70). In both cases, all the reconstructed images look pretty similar to the original ones. Although in image 'a' the reconstructions are much better than those of image 'b' because its resolution is higher and it is much more difficult to appreciate how it degrades. Looking closely at some details, e.g. the whiskers or the fur of the cat, it can be seen how degrades as the compression increases. Depending on the method, images can be degraded in different ways. In Fourier methods all images are degraded as if the number of pixels is reduced. It also appears as if grain appears and sometimes as if a

filter is applied. This type of degradation is progressive. Progressive degradation is of interest to us as opposed to breakage degradation. This is similar to the case where in structures we sometimes prefer deformation rather than fracture. Even if we apply high compression we can still recognize patterns and silhouettes in the image. In Figure (69), the JPEG method does not maintain the contrast as compression increases. This effect does not visibly happen in the other methods nor in the JPEG method of the Figure (70).

In Fgure (71) the MSSIM for the three methods different compression percentages can be seen. Using the MSSIM score, the best performing method is the JPEG followed closely by the DCT and FFT. The 8 by 8 block structure of the JPEG allows the code to be very fast and to be able to localize the frequencies as a STFT does. In Figure (71), the JPEG method is staggered because the threshold cutoff occurs every 8 by 8 block, unlike the other methods which apply to the entire image. This causes coefficients to be removed in a staggered manner. DCT performs better than FFT for two main reasons. The first, explained in the section 7.5, is that the DCT has one-half cycle basis functions that allow for gradual change from one side of the image to the other. The second is that, unlike the FFT, the DCT only obtains real numbers as a result, which allows it to reduce the compression by half.

a) [1600 x 1600] b) [360 x 360]

Figure 68 Original images used in this section with their sizes. Source image b): Stephen Hyde

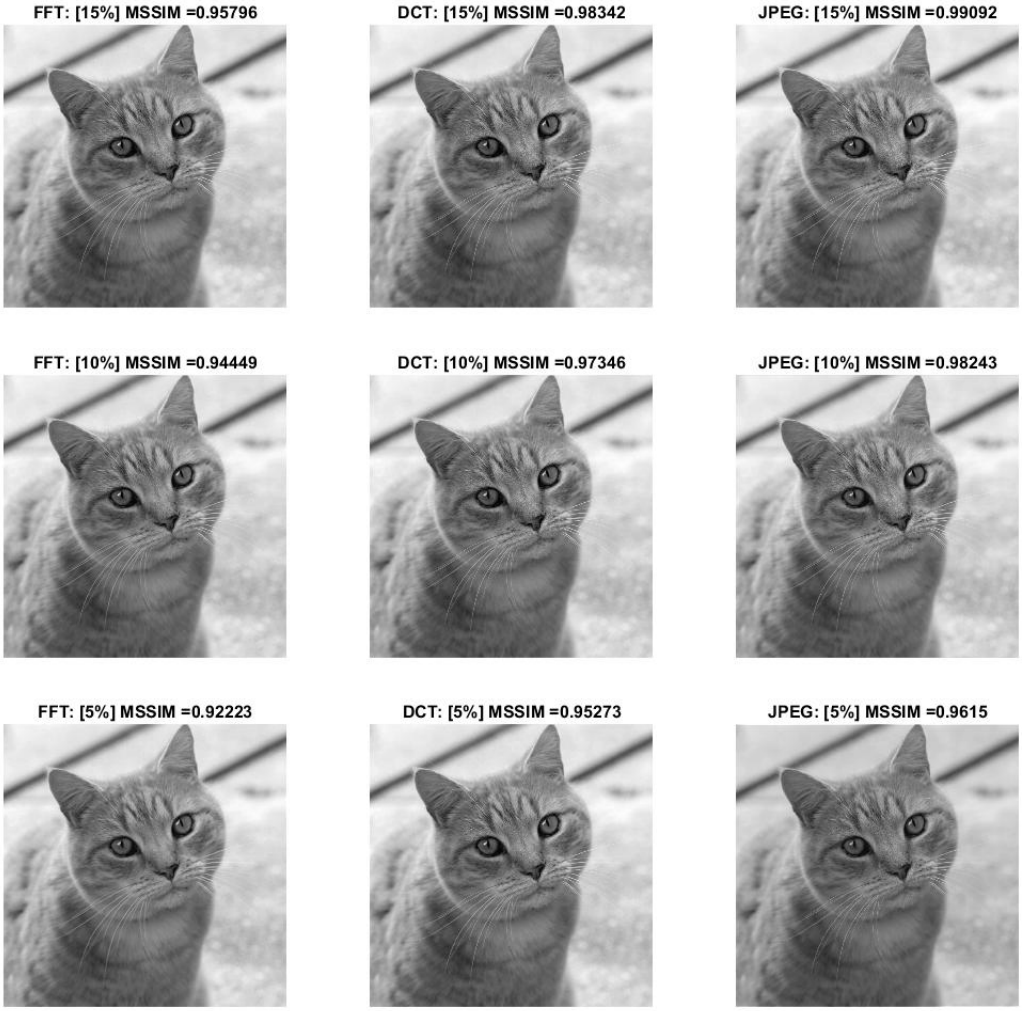


Figure 69 Comparison of Fourier transform methods (FFT, DCT and JPEG) at different compression rates (15, 10, and 5 percent) for image a) of Figure (68). Source own.

Figure 70 Comparison of Fourier transform methods (FFT, DCT and JPEG) at different compression rates (15, 10, and 5 percent) for image b) of Figure (68). Source own.

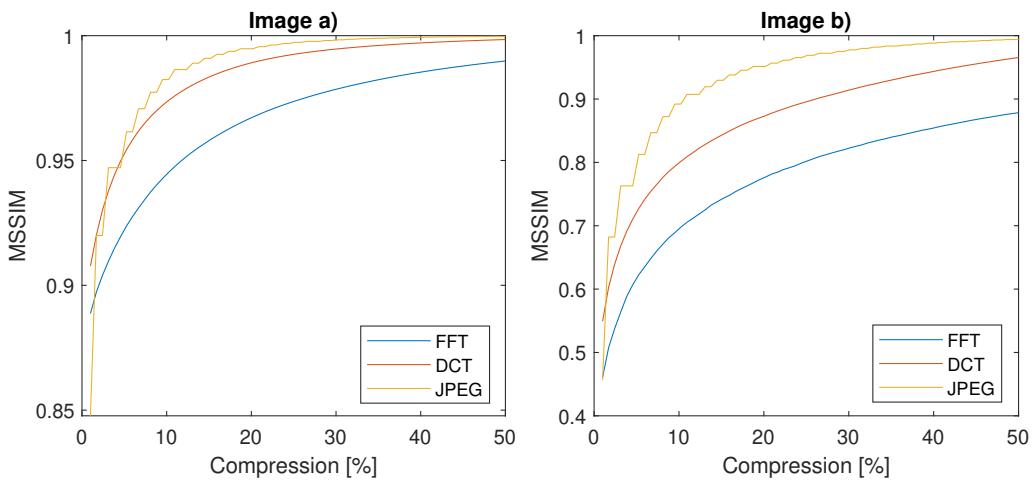


Figure 71 The MSSIM for the three Fourier compression methods at different compression percentages for images a) and b) of Figure (68). Source own.

10.2.1 Cut method for JPEG (threshold vs region)

In JPEG each 8-by-8 block can be cut for compression using the thresholding or region. The thresholding method only saves the most important coefficients while the region method saves a predefined region of the block. The region method can be accomplished by multiplying to each block a matrix called 'mask' where the number of ones depends on the amount of compression desired.

$$mask = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The results for each of the cut methods can be seen in Figure (72). As expected, the thresholding method obtains better results than the region method since stores only the most important information.

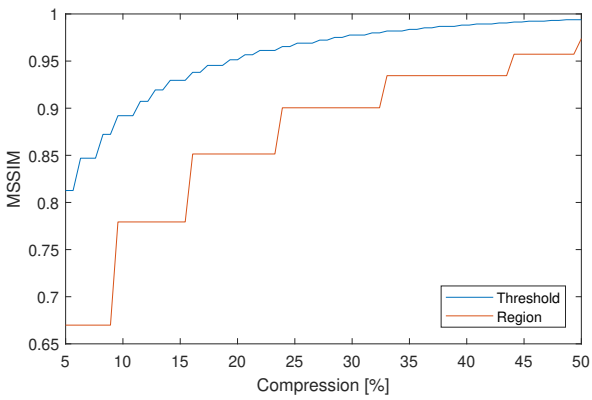


Figure 72 The MSSIM for the threshold and region cut methods at different compression percentages for image b) of Figure (68). Source own.

10.3 Comparison between wavelet transform methods

Following the same procure as in Fourier, in this section the performance of three different wavelet transform methods will be studied. The chosen methods are the Matlab wavelet decomposition (Mallat algorithm), the DWT, and the multiresolution lifting scheme. The wavelet that will be used is the most simple one, the Haar wavelet. The same image will be compressed via threshold at different compression rates for each method. The compression rate has been expressed as a percentage of the data stored in comparison with the data of the original image. The two images used for this test can be seen in Figure (68). The multilevel methods (Mallart and lifting scheme) are both used with 10 levels for image 'a' and 8 levels for image 'b'.

The results of this comparison can be seen in Figures (73) and (75). In both cases, the best performing algorithm is the multilevel lifting scheme followed closely by the Matlab decomposition

method. On the other hand, the DWT does not perform well under 25% of compression. In both cases at a compression of 20% black spots appear in some areas of the reconstructed image. This type of degradation is what I will call fracture degradation. This can be seen reflected in how the MSSIM score of the DWT in Figure (74) plummets after compression of 25%. To better visualize the MSSIM scores of the other two methods, the same plots but without the DWT can be seen in Figure (76). In this figure, it can be seen that the packet decomposition method has superior MSSIM scores for different compression rates.

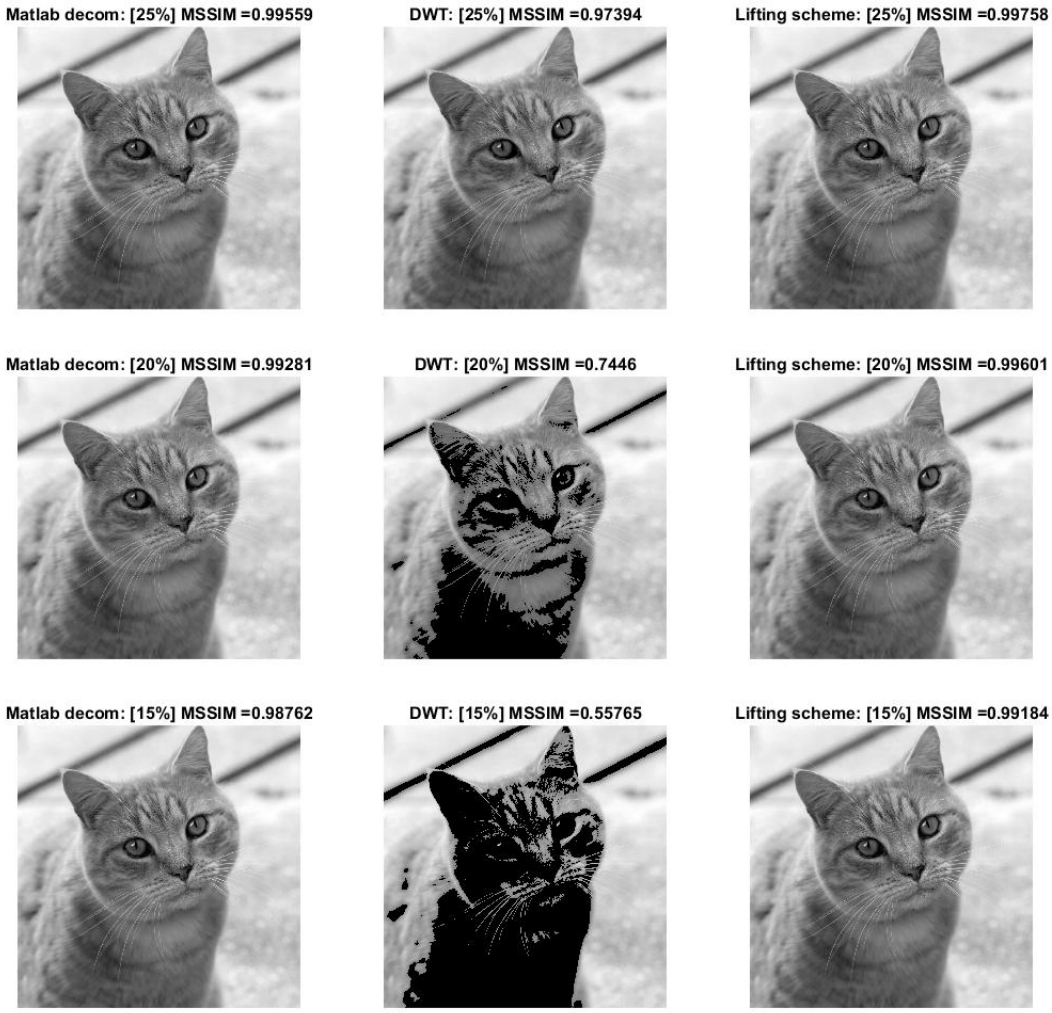


Figure 73 Comparison of wavelet transform methods (Matlab wavelet decomposition, DWT and lifting scheme) at different compression rates (25, 20, and 15 percent) for image a) of Figure (68). Source own.

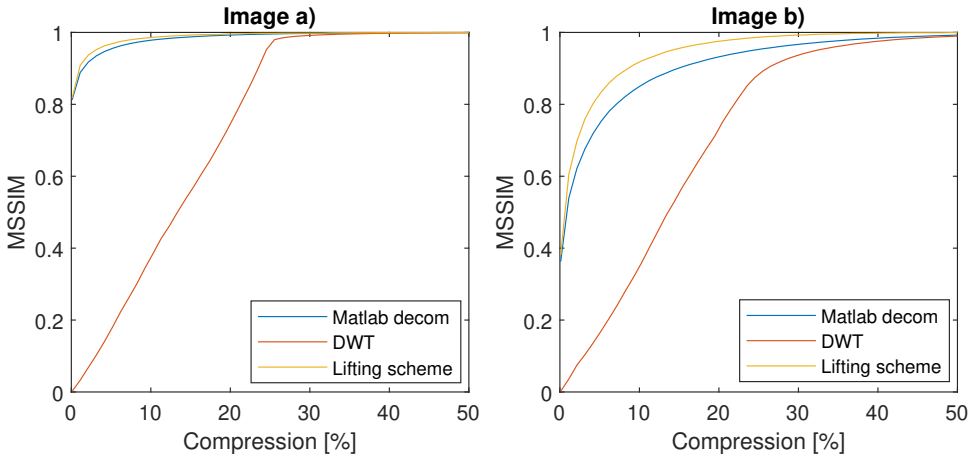


Figure 74 The MSSIM for three wavelet compression methods at different compression percentages for images a) and b) of Figure (68). Source own.

Figure 75 Comparison of wavelet transform methods (Matlab wavelet decomposition, DWT and lifting scheme) at different compression rates (25, 20, and 15 percent) for image b) of Figure (68). Source own.

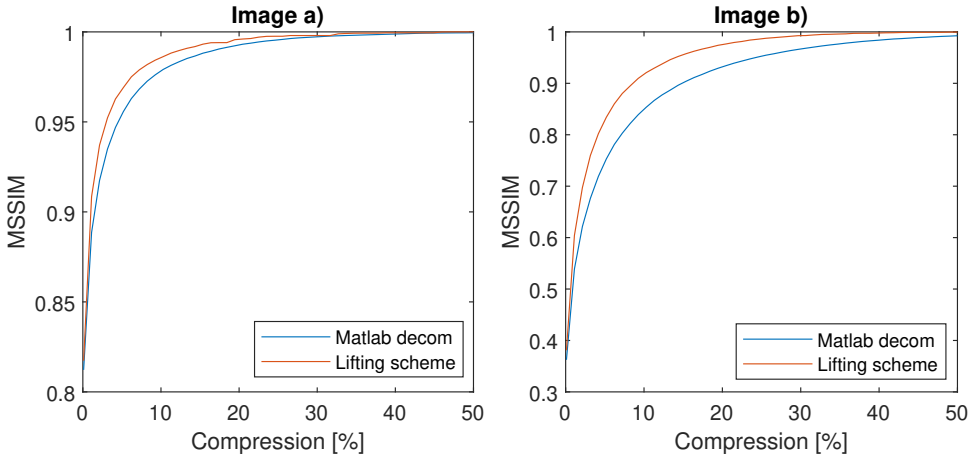


Figure 76 The MSSIM for the two best performing wavelet compression methods at different compression percentages for images a) and b) of Figure (68). Source own.

10.4 Fourier, Wavelet and PCA comparison

Finally, we will compare image compression using Fourier, wavelet, and PCA. For Fourier and wavelet, we will use the methods that have obtained the best results in the previous subsections. Therefore we will compare the JPEG method with threshold cutting, the multiresolution wavelet decomposition method using the lifting scheme, and the new method of this section, the PCA. For the wavelet multiresolution analysis, as in the wavelet subsection (section 10.3), the Haar wavelet will be used as the mother wavelet and the number of levels will be 10 for image 'a' and 8 for image 'b'.

The results of this comparison can be seen in Figures (77) and (79). In Figure (78) the MSSIM for the three methods at different compression rates can be seen. In both cases, the best-performing method is the wavelet followed closely by JPEG. PCA does not perform as well as the other two methods.

For now we have been relying on the MSSIM values which have been quite useful, but to really decide if one compression method is better than another must be done visually. In Figure (80) we can visually compare the JPEG and wavelet results of two images reconstructed with a similar MSSIM score. In these images the conflicting and expressive areas, the eye region and the facial fur, have been zoomed in. We can see that although wavelets has a slightly lower MSSIM score than JPEG, it is able to reproduce the original image much better, adapting much better to the abrupt jumps of the fur and the edges of the eyes. These jumps are difficult to mimic for Fourier because it only uses sine waves to reconstruct unlike wavelet which can choose its basis. This is related to the Gibbs phenomenon studied in section 7.1. In addition, in the JPEG method, near the eyes of the cat the 8 by 8 blocks can be observed.The wavelet method could be improved by using a more sophisticated wavelet, like the Daubechies. The results obtained agree with the advance of image compression technology where the image file format successor to JPEG is JPEG2000 which uses the wavelet transform to compress images.

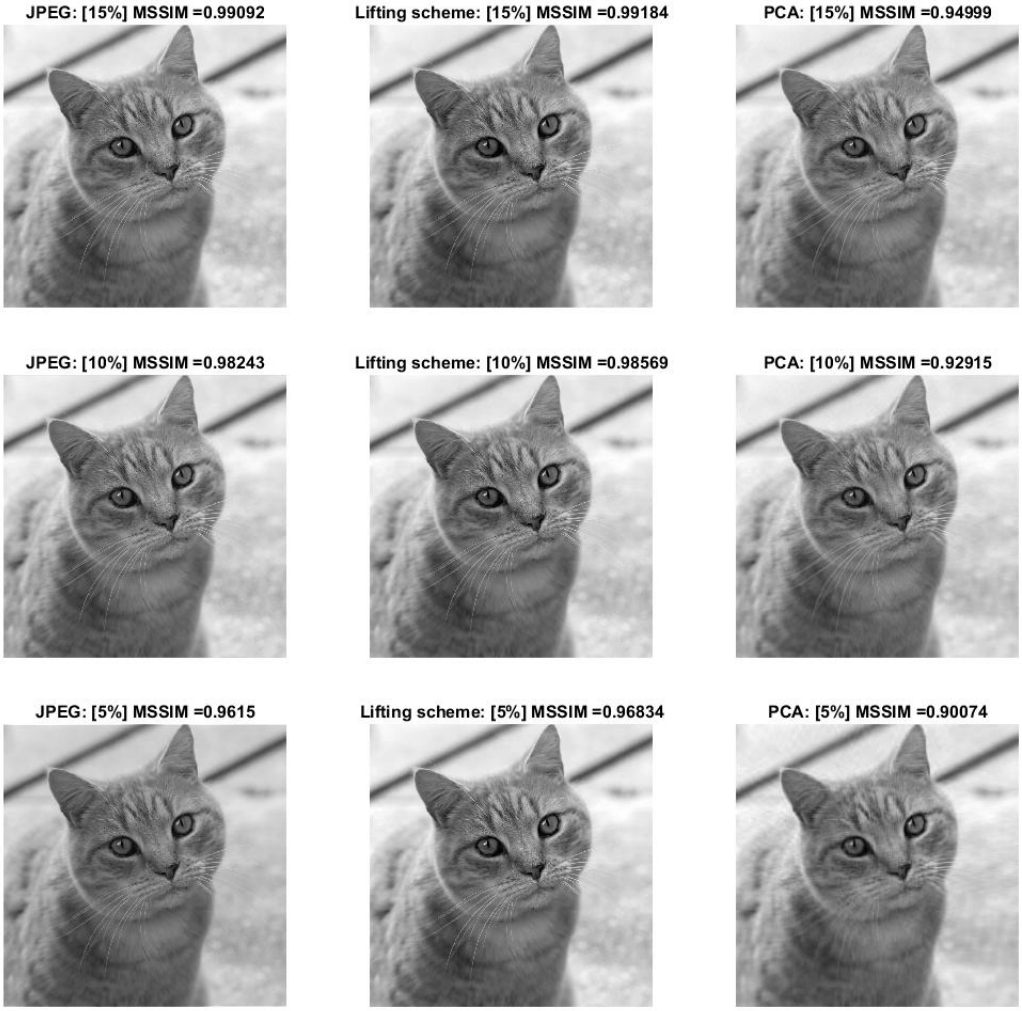


Figure 77 Comparison between image compression techniques (JPEG, multiresolution lifting scheme and PCA) at different compression rates (15, 10, and 5 percent) for image a) of Figure (68). Source own.

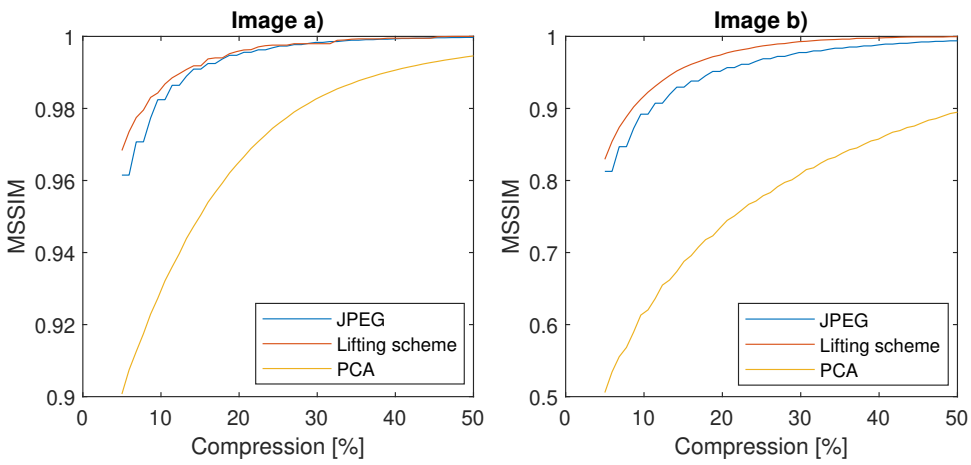


Figure 78 The MSSIM for the three methods (JPEG, multiresolution wavelet decomposition and PCA) at different compression percentages for images a) and b) of Figure (68). Source own.

Figure 79 Comparison between image compression techniques (JPEG, multiresolution lifting scheme and PCA) at different compression rates (15, 10, and 5 percent) for image b) of Figure (68). Source own.

Figure 80 Comparison between JPEG and multiresolution wavelet packet decomposition at similar MSZSIM scores for image a) of Figure (68)

10.5 Conclusions

The goal of this thesis was to understand how image compression works. To this end, we studied the mathematics behind the different compression methods and were able to apply the knowledge we learned to the development of a Matlab code capable of compressing and reconstructing audio (1D) and image (2D) signals through various algorithms. The code has been developed with the best possible balance between versatility and process automation while remaining user-friendly.

The Fourier transform proves to be a very useful tool for signal processing. The FFT allows this tool to be applied to large matrices without the number of operations being exorbitant, and by reducing the computation time. Applying the problem to digital images, optimizations such as block DCT can achieve very good compression with little computational power. The block division also allows for normalization of bases as well as localization of frequencies in time.

The wavelet transform is able to adapt Fourier analysis to more general orthogonal bases by using a multi-resolution decomposition. The wavelet transform can achieve better results than JPEG as well as to adapt better to some image patterns by adapting its basis to the image to be compressed. This new tool has evolved sufficiently to the point where it has detached itself completely from the Fourier world, creating what we call second-generation wavelet transforms. The wavelet transform is more complex than the Fourier transform and much more difficult to implement correctly. Also, at a high level of compression, there comes a point where the image degrades significantly. The degree of compression at which this degradation occurs varies from image to image and cannot be determined a priory. This is a major limitation of this method.

Finally, PCA is a useful tool for reducing the dimensionality of a data set by losing as little information as possible. Applied to the world of image processing, it can be useful in specific applications such as face recognition among others, but it is not capable of compressing images to as high a level as Fourier or wavelets.

The results of the comparison of the different compression methods studied are consistent with the historical evolution of image compression technology. In the first comparison, we can note the superiority of the JPEG method over other contemporary technologies. Subsequently, compression using the wavelet transform (JPEG200) can outperform JPEG, but with some drawbacks.

The wavelet sections have not been studied in as much detail as would be required to master this method. The creation of a custom wavelet or the compression ratio at which the reconstruction fails could be interesting sections to investigate. Other areas such as EM algorithms, learning kernels, or deep learning techniques for image compression could also be explored.

A GIFs

If you cannot view the gifs, try opening the document with Adobe Reader from Windows. If the gifs still cannot be visualized in this appendix, the gif frames will be displayed.

Figure 9

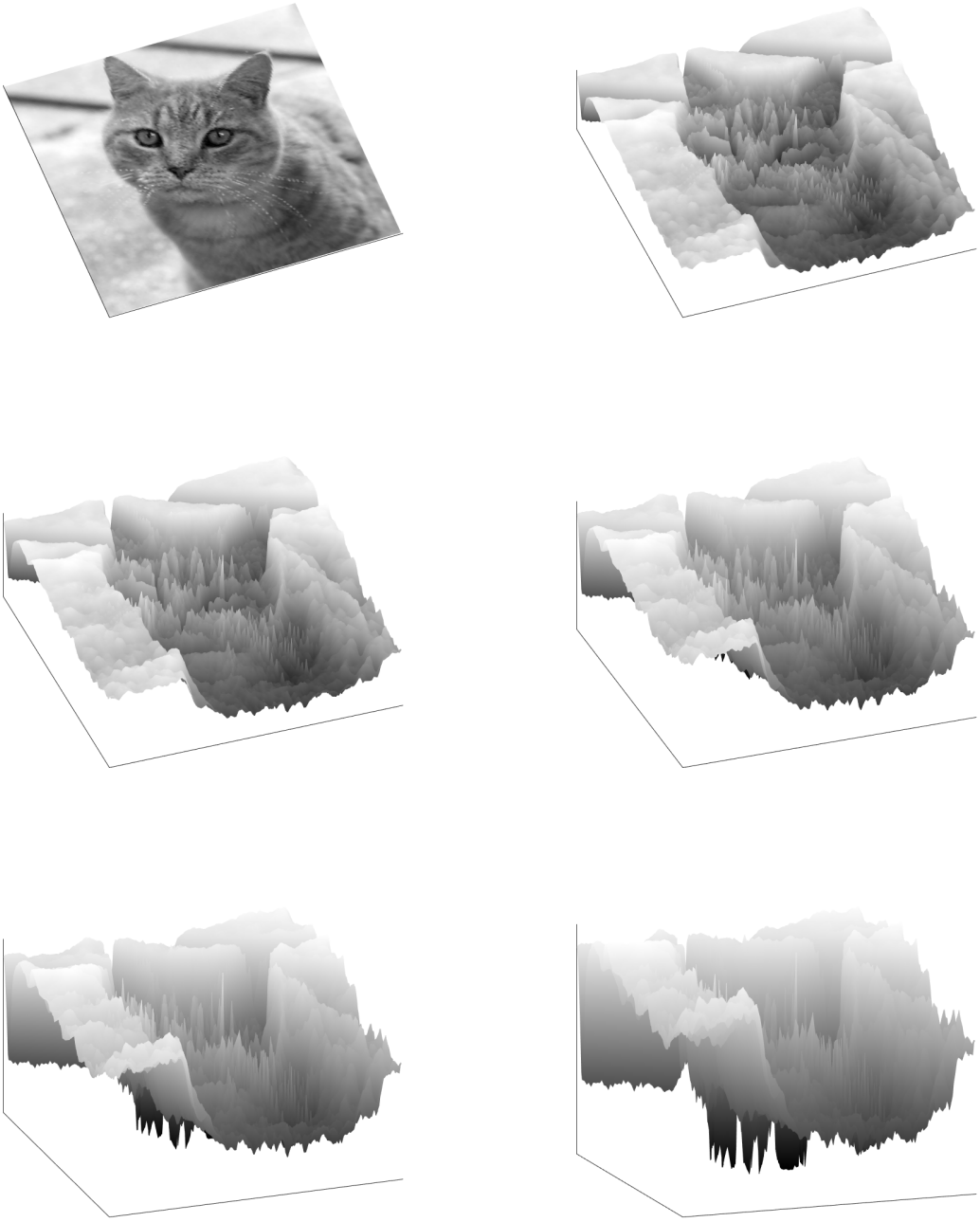


Figure 11

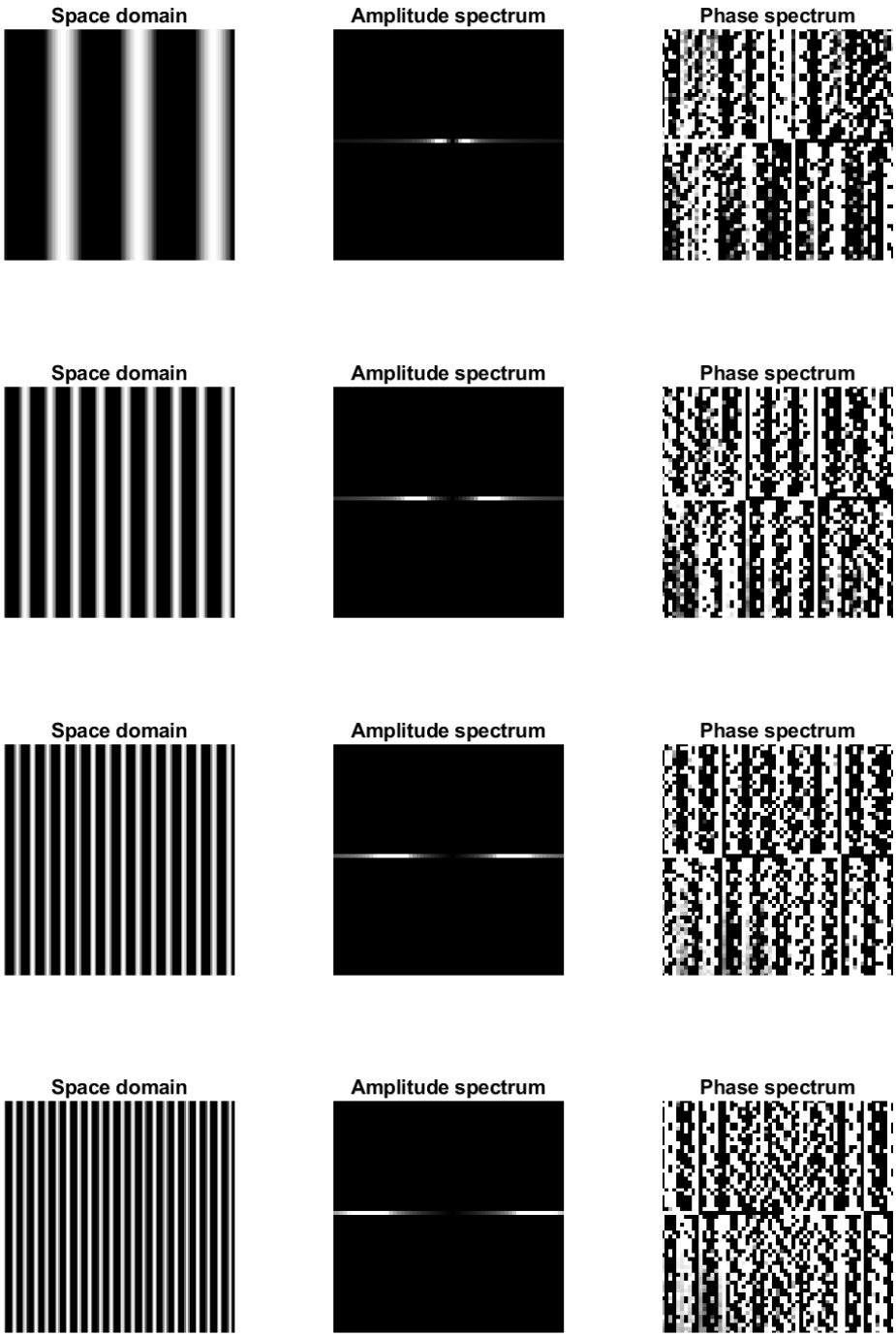

Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum

Figure 12

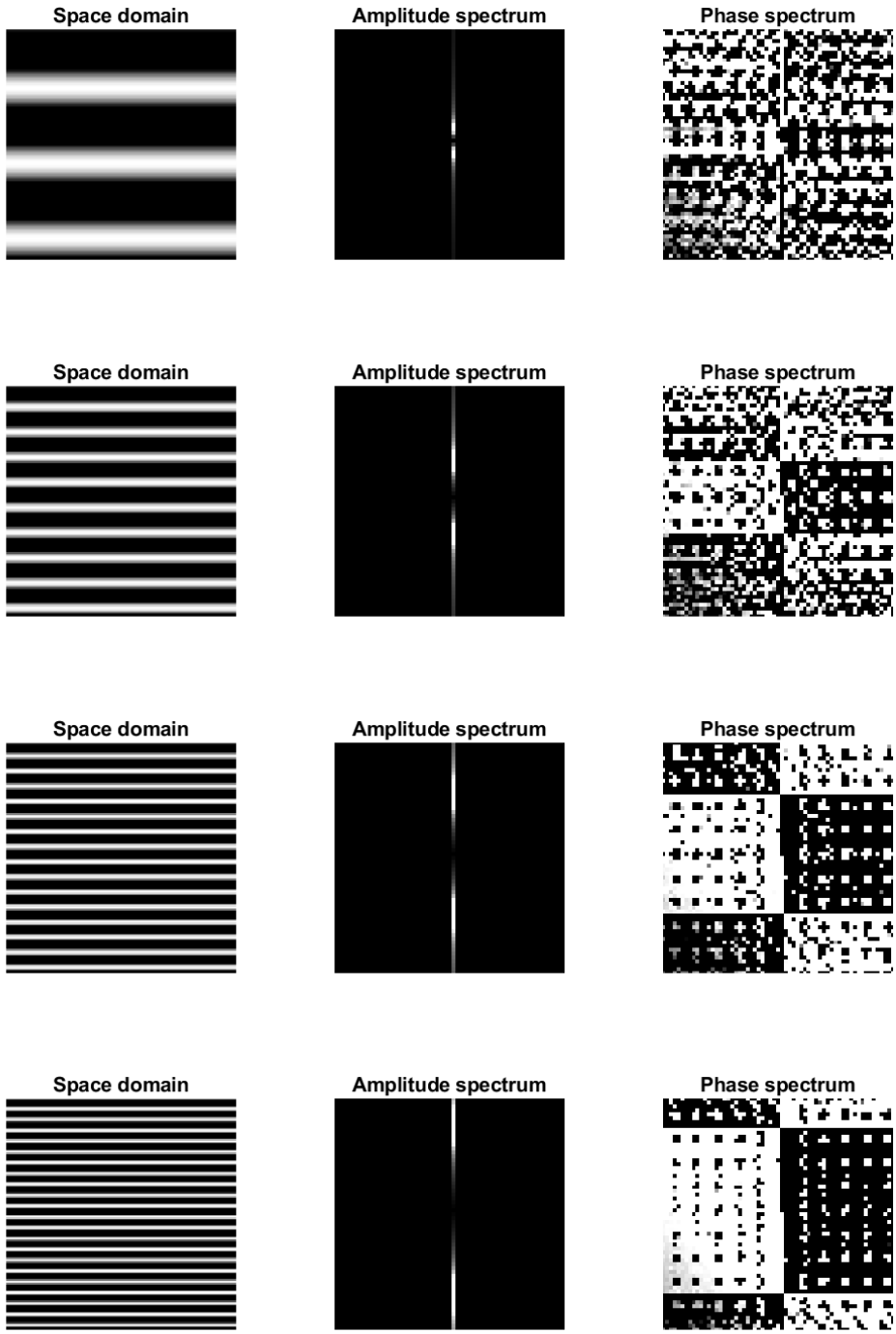

Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum

Figure 13

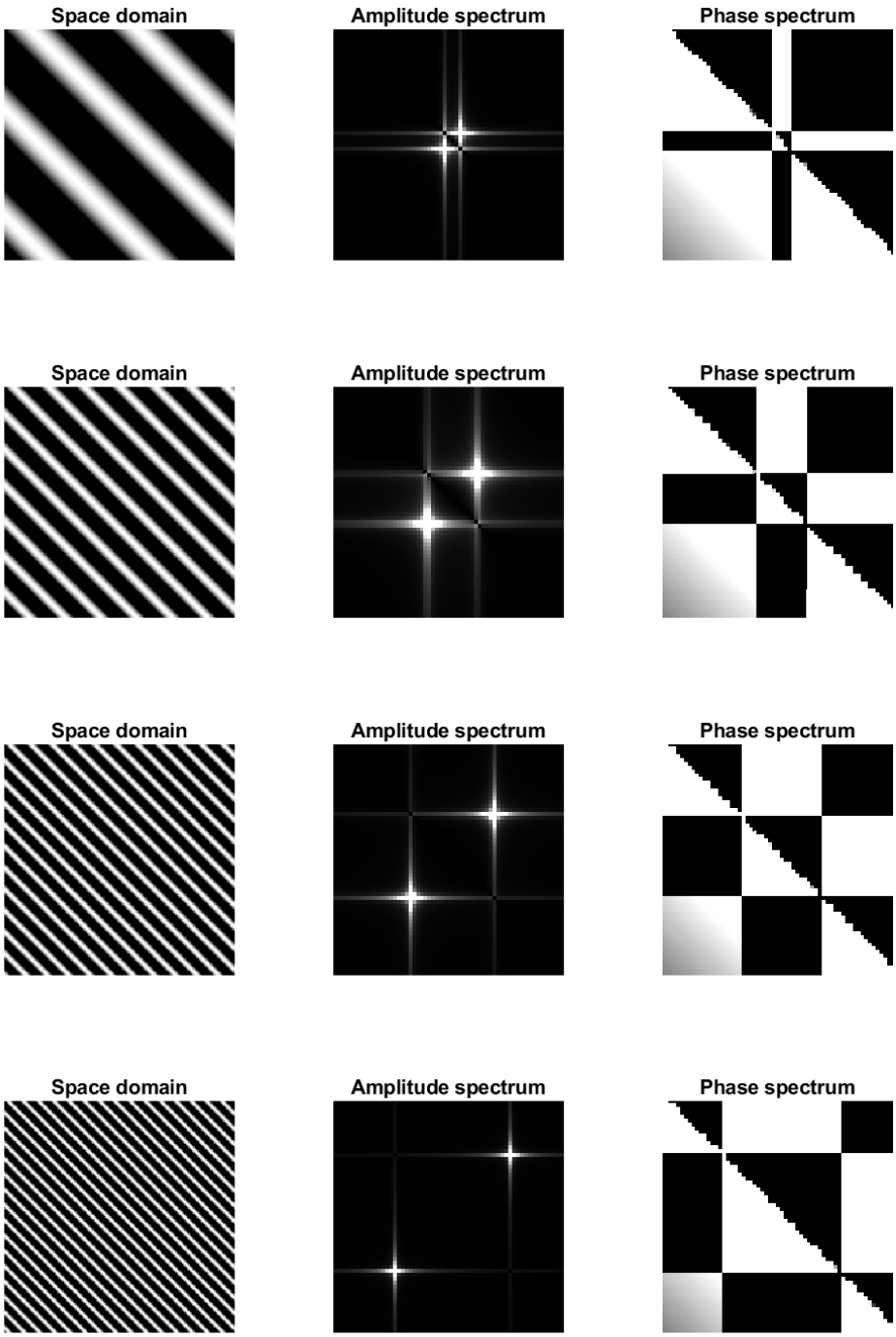

Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum

Figure 14

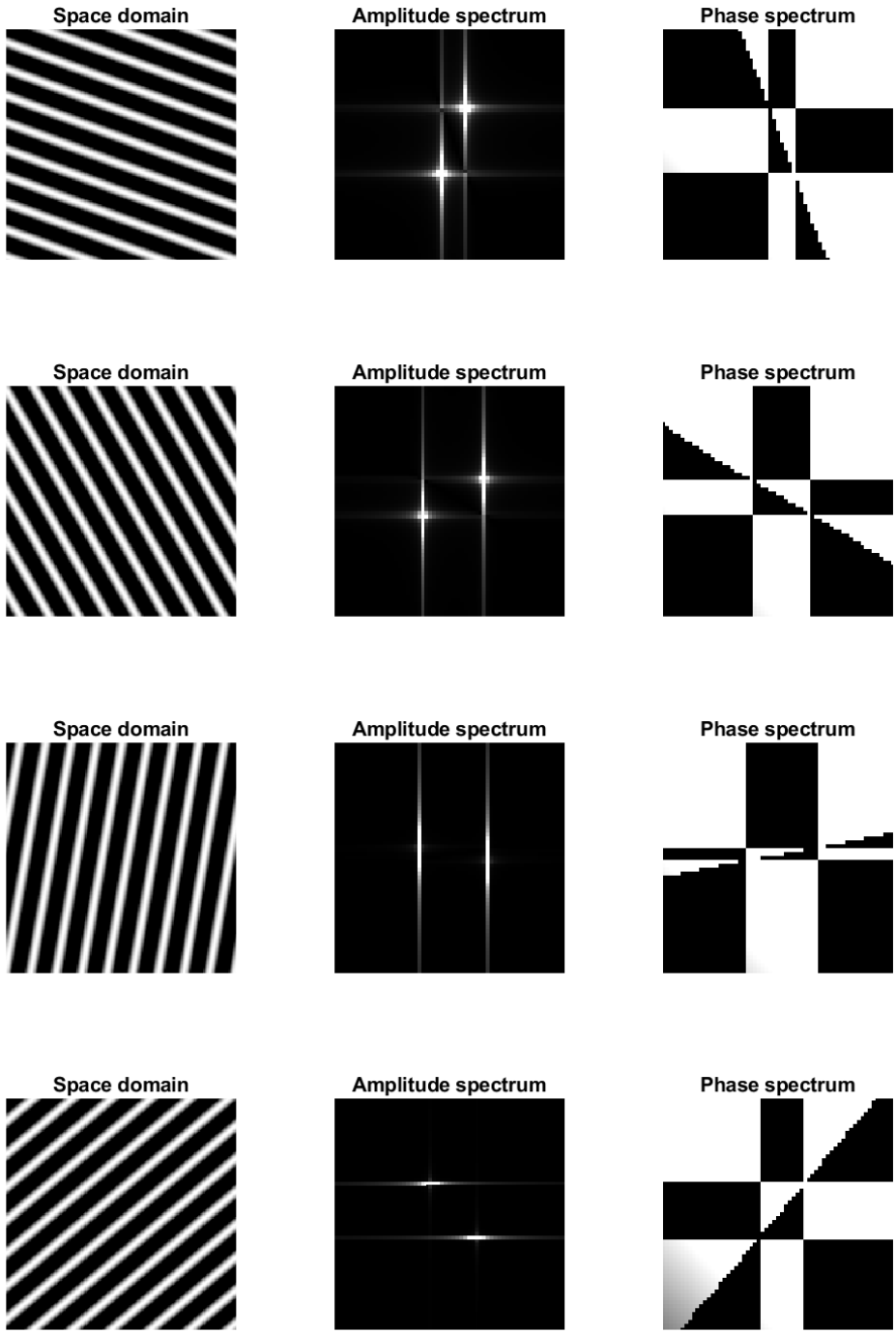
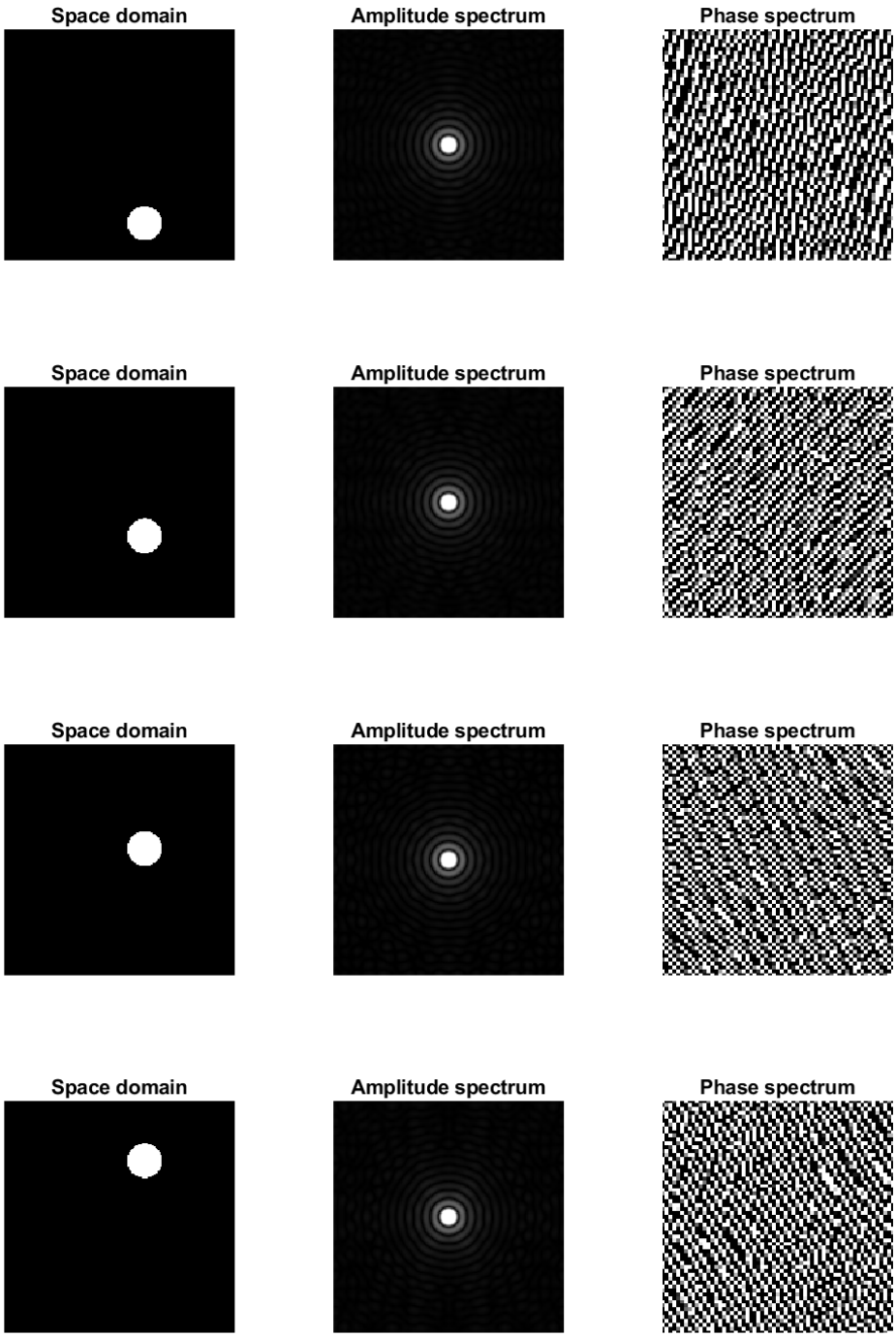

Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum

Figure 15


Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum
Space domain
Amplitude spectrum
Phase spectrum

References

[1] Paul S. Addison. "Introduction to redundancy rules: The continuous wavelet transform comes of age". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2126 (2018), p. 20170258. DOI: 10.1098/rsta.2017.0258.

[2] Paul S. Addison. *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in science, Engineering, Medicine and Finance*. CRC Press, Taylor; Francis Group, 2017.

[3] Nasir Ahmed, T. Raj Natarajan, and K. R. Rao. "Discrete Cosine Transform". In: *IEEE Transactions on Computers* 23 (1974), pp. 90–93.

[4] Ashok Ambardar. *Procesamiento de Señales Analogicas y digitales*. International Thomson Editores, S.A. de C.V., 2002.

[5] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. 1st. USA: Cambridge University Press, 2019. ISBN: 1108422098.

[6] Janko Calic. *Communication Systems*. Janko Calic, 2008. URL: https://cnx.org/contents/GXPlEBps@5.18:SP5IXuJr@1/Preface-for-Digital-Signal-Processing-A-User-s-Guide (visited on 06/07/2022).

[7] Siyuan Cao and Xiangpeng Chen. "The second-generation wavelet transform and its application in denoising of Seismic Data". In: *Applied Geophysics* 2.2 (2005), pp. 70–74. DOI: 10.1007/s11770-005-0034-4.

[8] CCITT. *Recommendation T.81: Terminal Equipment and Protocols for Telematic Services : Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines*. 1992. URL: https://www.w3.org/Graphics/JPEG/itu-t81.pdf (visited on 05/14/2022).

[9] E. Chandrasekhar, V. Dimri, and Vikram Gadre. *Wavelets and Fractals in Earth System Sciences*. Nov. 2013. ISBN: 13: 9781466553590. DOI: 10.1201/b16046.

[10] R.L. Claypoole et al. "Nonlinear wavelet transforms for image coding via lifting". In: *IEEE Transactions on Image Processing* 12.12 (2003), pp. 1449–1459. DOI: 10.1109/TIP.2003.817237.

[11] James W. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19 (1965), pp. 297–301.

[12] Arie Dickman. *Verified signal processing algorithms in Matlab and C*. Springer Nature, 2022.

[13] *Discrete Cosine Transform*. URL: https://es.mathworks.com/help/images/discrete-cosine-transform.html (visited on 05/10/2022).

[14] Fir0002. *Common housecat*. July 2005. URL: https://commons.wikimedia.org/wiki/File:Cat03.jpg.

[15] Jean Baptiste Joseph Fourier. *The Analytical Theory of Heat*. Ed. by AlexanderTranslator Freeman. Cambridge Library Collection - Mathematics. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511693205.

[16] Matteo Frigo and Steven G. Johnson. "The Fastest Fourier Transform in the West (MIT-LCS-TR-728)". In: 1997.

[17] D. Gabor. "Theory of communication". In: *J. Inst. Electr. Engineering* (1946), pp. 429–457.

[18] Robert X. Gao and Ruqiang Yan. *Wavelets theory and applications for manufacturing*. Springer, 2011.

[19] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 2018.

[20] Amara Graps. ""An Introduction to Wavelets"". In: *IEEE Comp. Sci. Engi.* 2 (Feb. 1995), pp. 50–61. DOI: 10.1109/99.388960.

[21] Richard F. Haines and Sherry L. Chuang. "The effects of video compression on acceptability of images for monitoring life sciences experiments". In: 1992. URL: https://ntrs.nasa.gov/citations/19920024689 (visited on 06/07/2022).

[22] Yuheng He et al. "A Matrix-Vector Based Approach to FFT Implementations". In: Jan. 2010, pp. 490–494. DOI: 10.1109/ISSPIT.2009.5407501.

[23] Emmanuel C. Ifeachor and Barrie W. Jervis. *Digital Signal Processing: A practical approach*. Prentice Hall, 2011.

[24] W. Kinsner. "Compression and its metrics for multimedia". In: *Proceedings First IEEE International Conference on Cognitive Informatics*. 2002, pp. 107–121. DOI: 10.1109/COGINF.2002.1039289.

[25] Charles Van Loan. *Computational frameworks for the fast fourier transform*. Society for Industrial and Applied Mathematics, 1992.

[26] S.G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 674–693. DOI: 10.1109/34.192463.

[27] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. 3rd. USA: Academic Press, Inc., 2008. ISBN: 0123743702.

[28] Kevin P. Murphy. *Probabilistic machine learning: An introduction*. The MIT Press, 2022.

[29] Nina Pantic. *How many photos will be taken in 2021?* Feb. 2022. URL: https://blog.mylio.com/how-many-photos-will-be-taken-in-2021-stats/ (visited on 03/14/2022).

[30] Grant Sanderson. *But what is the Fourier transform? A visualintroduction*. Jan. 2018. URL: https://www.3blue1brown.com/lessons/fourier-transforms (visited on 02/14/2022).

[31] Fadil Santosa. *Second Generation Wavelets: Theory and Application*. May 2003. URL: https://web.archive.org/web/20150913001138/http://www.ima.umn.edu/industrial/97_98/sweldens/fourth.html (visited on 05/14/2022).

[32] Adam Siembida. *How to compute the IFFT using only the forward FFT*. Mar. 2010. URL: https://adamsiembida.com/how-to-compute-the-ifft-using-only-the-forward-fft/ (visited on 05/05/2022).

[33] L. Sirovich and M. Kirby. "Low-dimensional procedure for the characterization of human faces". In: *J. Opt. Soc. Am. A* 4.3 (Mar. 1987), pp. 519–524. DOI: 10.1364/JOSAA.4.000519. URL: http://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-3-519.

[34] Steven W. Smith. *The scientist and engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.

[35] Charles S. Swartz. "Compression systems and standards". In: *Understanding Digital Cinema: A professional handbook*. Elsevier, 2005, pp. 146–147.

[36] Wim Sweldens. "The lifting scheme: A construction of second generation wavelets". In: *SIAM Journal on Mathematical Analysis* 29.2 (1998), pp. 511–546. DOI: 10.1137/s0036141095289051.

[37] M.A. Turk and A.P. Pentland. "Face recognition using eigenfaces". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 586–591. DOI: 10.1109/CVPR.1991.139758.

[38] Z. Wang et al. "Image quality assessment: From error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/tip.2003.819861.

[39] Andrew Zisserman. *Lecture 2: 2D Fourier transforms and applications*. 2014. URL: https://www.robots.ox.ac.uk/~az/lectures/ia/.