Dissertation

Graph Spectral Image Processing over Adaptive Triangulations

Universität Hamburg
Department of Mathematics

Dissertation with the aim of achieving a doctoral degree at the
Faculty of Mathematics, Informatics and Natural Sciences

submitted by
Niklas Wagner

Hamburg, 2021

Als Dissertation angenommen vom Fachbereich Mathematik der Universität Hamburg auf Grund der Gutachten von:

Prof. Dr. Armin Iske
Prof. Dr. Jörn Behrens

Datum der Disputation: 03.09.2021

Acknowledgements

First and foremost I want to thank my supervisor, Prof. Dr. Armin Iske, for his support, guidance, encouragement and patience not only throughout my PhD project, but also during my Bachelor and Masters thesis. Whenever I needed his help he had an open ear and great advice!

Besides my supervisor, I want to thank Prof. Dr. Jörn Behrens for taking the time and effort of reading and evaluating my thesis.

Then I want to thank my colleagues and fellow PhD students, especially my working group as well as the SIAM & GAMM Chapter Hamburg. I really enjoyed the time together at the Geomatikum, be it countless trips to the Mensa, coffee breaks, Officer meetings, in-depth discussions of the previous NFL GameDay and much more.

I want to thank my friends and family for enduring this long process with me, offering support, encouragement and also distraction whenever I needed them.

My deepest thanks goes to my wife Doreen for everything.

Abstract

With the rapidly advancing developement of smartphones with integrated cameras, digital image compression is still a relevant and ongoing research topic. Conventional schemes, such as JPEG or JPEG 2000, rely on fixed transforms over regular grids. Opposed to this, adaptive thinning is an image compression scheme that reconstructs an image with linear splines over the anisotropic Delaunay triangulation of a set of adaptively chosen significant pixels.

The main objective of this thesis is to improve the performance of this scheme, especially on textured images. To this end, we propose a post-processing procedure that improves selected regions of the reconstruction by utilizing graph signal processing as a tool to define frequency spectra on irregular, triangular image domains.

Our approach designs adaptive graphs to exploit signal smoothness of graph signals. To this end, significant triangular image blocks are classified via the structure tensor based on their textural content. Graphs are constructed that promote sparseness of the graph Fourier spectrum. This is achieved either by graph learning methods or a discrete weight model, where the edge weight is chosen optimally.

Based on the constructed graphs, the signal is transformed via the graph Fourier transform to the graph spectral domain, where thresholding and quantization can be performed efficiently.

Finally, we show how to implement this scheme, focusing on the transformation and quantization. We compare it with the adaptive thinning scheme on geometric and natural images.

Contents

Chapter 1

Introduction

Images are omnipresent in this day and age. Due to the rising popularity of mobile smartphones with modern digital cameras, everyone can capture images wherever and whenever they want. And due to companies such as Facebook, Instagram or Snapchat encouraging their users to share their private images, there is no shortage of incentives. Social networks' prime business area is based on sharing images, and many companies use this as marketing strategies. Therefore, it is not surprising that the amount of images taken anually is growing rapidly. In the year 2000, when film photography was still prevalent, only an estimated 80 billion photos were taken globally [42]. The explosion of digital photography and smartphones led to over 1.4 trillion photos being taken in 2020. As a consequence, an estimated 7.4 trillion images were stored 2020, and this number is expected to grow to 9.3 trillion in 2022.

The transition from analog to digital images gave rise to the research on *digital image compression*. A digital image is a collection of intensity values given on a discrete set of pixels. Storing a digital image naively with eight bits per pixel, only three 1280×720 grayscale images would fit on a 2.88 MB floppy disc, the prevalent storage medium just 30 years ago. Colour images consist of three colour channels, requiring 24 bits per pixel, so only one 1280×720 colour image fits on a floppy disc.

Even though the data storage technology has rapidly advanced in the recent decades and storage mediums are several magnitudes larger, so did the size of images. Compact smartphones have cameras capable of taking images with 108 Megapixels, leading to a naive storage cost of 324 MB. Those images do not only have to be stored, but also transmitted via a mobile internet connection. Therefore, image compression is still a relevant and ongoing research topic.

The goal of image compression is to store an image in a more compact form, i.e. a representation that requires fewer bits for encoding than the original image. This is possible for most images, because their original representation contains a lot of redundant data. Most images are not a collection of arbitrary intensity values, but rather contain some sort of structure. As such, neighbouring pixels are usually correlated. Ideally, an image compression technique removes redundant or irrelevant information and efficiently encodes the remaining information. In practice, it is often necessary to remove both nonredundant and relevant data to achieve the desired compression ratio.

Image transform coding is the most commonly used image compression technique. During encoding the raw data is transformed to decorrelate it and efficiently extract the relevant information. The most widely used transform in image compression is the *discrete cosine transform*

(DCT), introduced by Ahmed, Natarajan and Rao in 1974, see [2]. It is a Fourier-related transform that is similar to the discrete Fourier transform but only uses real numbers. With the DCT as basis, the JPEG compression standard was developed by the Joint Photographic Experts Group and introduced in 1992, see [89], and is still the prevalent digital image format.

As a Fourier-related transform, it transfers an image to the frequency domain, where most frequencies are irrelevant for the image at hand. But it also results in a representation that has no localization in the spatial domain. Thus, the JPEG committee developed the JPEG 2000 image coding system [9], which was introduced in 2000. It is based on the *discrete wavelet transform* (DWT), that represents a signal via an orthonormal basis generated by a wavelet, see [12]. The orthonormal basis is constructed by dilation and translation of a mother wavelet, resulting in a *multiresolution analysis*, which was introduced in this context by Mallat in 1989, see [63]. In the following years several multiscale methods have been proposed, such as *wedgelets* [25], *curvelets* [6], *contourlets* [24], *bandelets* [58], *shearlets* [57] and the *easy path wavelet transform (EPWT)* [71],[72]. The EPWT constructs a graph along the DWT coefficients, so that there is a strong correlation between neighbouring data points. It therefore links nicely to the graph techniques introduced below.

A method to decorrelate an image without transforming it is given by the *adaptive thinning* (AT) algorithm, introduced by Demaret, Dyn and Iske in [21]. It constructs a representative, sparse subset of *significant pixels* via a thinning algorithm. Based on the luminances of these carefully selected pixels, the remaining images are approximated over a *Delaunay triangulation*. As a technique to approximate a large set of bivarate scattered data, the method was first created in [26] for terrain modelling, before being extended to image compression in [15]. Several advancements have been made, such as an improved pixel selection process in [16] and [20], post-processing techniques in [21] and [17], and efficient contextual coding in [22]. In [19], optimal N-term approximation rates were shown for the relevant classes of piecewise linear α-horizon functions and regular functions. [23] generalized the concept of adaptive thinning to approximate video data. Optimal N-term approximation rates for trivariate α-horizon functions were proven in [50].

In recent years, a new field of signal processing garnered much attention: *graph signal processing* (GSP) [80], [77]. Opposed to conventional signal processing, where signals lie on a regular grid in the temporal or spatial domain, *graph signals* are given on irregular domains. Values of the graph signals are defined on the vertices of a weighted graph, where the graph weights represent the pairwise relationships between those data points. Such a flexible representation improves on fixed transforms like the DCT by defining transforms that are adapted to the actual signal. Based on spectral graph theory [10], graphs are leveraged as tools to extend techniques and intuitions of conventional signal processing to the graph domain. To this end, Hammond, Vandergheynst and Gribonval introduced the *graph Fourier transform* (GFT) in [41]. It allows structure aware transform coding, which has been applied to a wide variety of applications such as *sensor networks* [29], *biological networks* [46], *3D point cloud processing* [86] and *machine learning* [90], [14].

Turning to image compression, the GFT is well suited for images that contain sharp boundaries, which are not reconstructed well by the DCT. A graph spectral transform on a suitable graph, i.e. one that captures the image structure, leads to a sparse representation. The challenge is to design a suitable graph and store it efficiently. Several methods for piecewise smooth and natural images have been proposed, see for example [79], [45], [69], [75], [33].

In this thesis we combine the AT algorithm with transform coding techniques from GSP. We aim to balance the good reconstruction of geometrical features by AT with a good reconstruction of textures in the graph spectral domain. Based on the Delaunay triangulation output by the AT algorithm, we add textures over a set of *significant triangles*. Those triangles are usually heavily textured and we capture these textures with an adaptive weighted matrix. We show how to construct these adaptive matrices so that they both represent the textures and are efficiently encoded. The main part of this thesis is organized in five chapters as follows.

Chapter 2 introduces the basic concepts of image compression. We begin with an overview of the basic definitions for image processing and introduce measurements for the quality of image compression schemes. Furthermore, the basic principles of transform coding and quantization are presented. Having laid the foundations, we introduce three popular discrete transforms. First, the Karhunen-Loève transform (KLT) is introduced as the transform with optimal decorrelative properties for a signal with a known correlation model. Next, the two-dimensional DCT is presented along with the JPEG coding scheme. Finally, we introduce the multiresolution analysis as an important tool in the JPEG-2000 scheme.

In Chapter 3 we introduce the AT method. To this end, we show how to represent an image via linear splines over a unique Delaunay triangulation. Next, we assign a significance to each pixel in the thinning process. Based on these significances, a set of most significant pixels is determined to represent an image most effective. The last two sections present post-processing steps and the final algorithm.

Relevant techniques from graph signal processing are presented in Chapter 4. Introducing the basic definitions of weighted graphs and graph signals in the first two sections, we then turn to generalizing the Fourier transform to graphs. Analogously to conventional signal processing, a sense of smoothness is presented for graph signals and linked to eigenvectors of the *graph Laplacian matrix*. This lets us define the GFT in the last section, that equivalently represents a graph signal in the vertex and graph spectral domain.

Having introduced the necessary building blocks, we propose our method in Chapter 5. We start with describing related work on graph spectral image compression found in the literature in Section 5.1 and explain similarities and differences to our approach. Section 5.2 motivates the actual signal to be encoded. It is given as the difference between the original image and the AT reconstruction, which preserves the good representation of sharp edges. The basic compression scheme for single triangular image blocks is presented in Section 5.3. A graph is constructed on the pixels of the image block with a fixed edge template. The signal is then transformed to the graph spectral domain, where filtering methods may be applied. As we aim to capture

textures with adaptive graphs, the impact of signal smoothness is demonstrated in Section 5.4. Furthermore, we explain the basic approach via graph learning and discrete weight models. Since adaptive graphs are only beneficial on textured blocks, we introduce the *block structure tensor* as a tool to classify each block in Section 5.5. Depending on its characteristics, each textured block is classified as isotropic or anisotropic. Section 5.6 explains how to construct weighted graphs on blocks with anisotropic textures. They are summarized by a dominant gradient, which is the basis for an adaptive graph template construction similar to [75]. Blocks with isotropic textures are not so easily summarized. Therefore we introduce two methods for graph construction on them in Section 5.7. The first one adapts a graph learning algorithm proposed in [33] which is optimized for high bit-rates. Next, we propose a method with a discrete weight model that is suited for low bit-rates. Finally, in Section 5.8, we generalize the approach in [45] and compute the optimal edge weight for weakly correlated pixels in a textured signal.

In Chapter 6 we provide experimental results. In Section 6.1 the experimental setup is explained in-depth, along with the choice of parameters in the implementation. Section 6.2 shows the algorithm applied to several geometrical and natural images and compares its performance to that of the original AT algorithm.

Chapter 2

Image Compression via Transform Coding

In this chapter we give a basic overview of the fundamental definitions and techniques of image processing and introduce transform coding image compression schemes.

Image transform coding is the most used image compression technique. It is a reversible process that transforms the raw data to decorrelate it and efficiently extract the relevant information, see [74]. A typical compression framework using image transform coding is depicted in Figure 2.1.

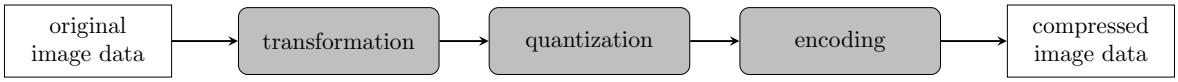


Figure 2.1: Typical transform coding framework.

The *transformation* expands an image signal in terms of a set of basis images and provides coefficients which quantify their respective contribution. These basis images are given by an orthonormal matrix and the coefficients are utilized in further processing steps. Choosing a transformation that efficiently decorrelates the data is key for an efficient compression.

After decorrelating the signal, *quantization* is applied to reduce the accuracy of the coefficients to a discrete set suited for encoding. Since the transformation is a reversible process, losses in quality are introduced in this step.

The quantized coefficient are then encoded efficiently. This is usually done by *entropy encoding*, for example using Huffman code, see [47]. Basically, more probably coefficients are given shorter codewords. Further compression can be achieved by thresholding less important coefficients at the cost of further distortion.

We start this chapter by introducing the basic definitions for digital image processing, found for example in [35] and [52]. Afterwards, we introduce the Karhunen-Loève transform (KLT) [74], which optimizes the decorrelation of a random signal whose satistics are known probabilistically. As we will see, the KLT requires the eigenvectors of the covariance matrix of the signal and thus has a very large overhead. This, as well as high computational costs, means that it is rarely used in practical applications. Various alternative, nonadaptive transforms have been utilized for image compression, such as the discrete cosine transform [89] or the discrete wavelet transform [9], which we will discuss in the last two sections.

2.1 Basics of Image Processing

In this thesis, unless otherwise stated, we operate on *digital greyscale images*. That is, we consider an image $\mathbf{I}$ as a two dimensional collection of greyscale values, i.e.

$$\mathbf{I}(x_1, x_2), \quad x_1 \in \{1, \dots, X_1\},\ x_2 \in \{1, \dots, X_2\}.$$

X_1 and X_2 are finite natural numbers and form the extent of the vertical and horizontal directions respectively. We say an image has a *resolution* of $X_1 \times X_2$. Each index $(x_1, x_2) \in \{1, \dots, X_1\} \times \{1, \dots, X_2\}$ represents a separate *picture element*, also known as *pixel*. All pixels together form the set $\mathrm{P} = \{(x_1, x_2) \in \mathbb{N} \times \mathbb{N} : 1 \leq x_1 \leq X_1,\ 1 \leq x_2 \leq X_2\}$. The greyscale value is also known as *luminance* and is a representation of the intensity (brightness) at its respective pixel. It is taken from a discrete set, usually an r-bit unsigned integer, i.e.

$$\mathbf{I}(x_1, x_2) \in \{0, 1, \dots, 2^r - 1\} \tag{2.1}$$

for $r \in \mathbb{N}^+$. Hence, a digital image can be represented as an element $\mathbf{I} \in \{0, 1, \dots, 2^r - 1\}^{\mathrm{P}}$ for a given r-bit length. $\mathbf{I}(x_1, x_2) = 0$ means no intensity at pixel (x_1, x_2), so it will be displayed black, whereas $\mathbf{I}(x_1, x_2) = 2^r - 1$ means full intensity and it will be displayed as white.

As stated above, the goal of image compression is to store an image in a more compact form. There are two main approaches: lossless and lossy image compression. Losless compression algorithms compress the file size without any loss to the image quality. This is achieved by coding the existing data efficiently. Lossy compression on the other hand removes data, which results in an approximation $\widetilde{\mathbf{I}}$ of the original image $\mathbf{I}$. Consequently, while losless compression has a higher quality, lossy compression is far more effective at reducing the representation cost.

In this thesis we focus on lossy compression schemes that do not change the resolution or color depth. That is, the set of pixels P and the r-bit length stays the same, so that $\widetilde{\mathbf{I}} \in \{0, 1, \dots, 2^r - 1\}^{\mathrm{P}}$. From here on out we will assume the standard r-bit length of $r = 8$, which yields an intensity range of $\{0, \dots, 255\}$.

Transform Coding

Transforms form the basis of many lossy image compression schemes. A well chosen transform removes redundancy in the data and compacts the information. This allows the least distortion when a loss of image quality arises during the quantization step. We focus on one-dimensional discrete transforms, as digital images are given on a discrete set of pixels and most image transforms are separable. That is, most two-dimensional transforms on signals such as images are realised by the tensor product of one-dimensional transforms, see [52].

Definition 2.1 *Given a discrete signal* $\mathbf{f} \in \mathbb{R}^N$*, a* discrete transform *is defined via a* transformation matrix $\mathbf{T} = [\mathbf{T}(k,l)]_{1\leq k\leq M,\ 1\leq l\leq N} \in \mathbb{R}^{M\times N}$ *by*

$$\mathbf{y} = \mathbf{T}\mathbf{f}. \tag{2.2}$$

For this thesis we only consider non expansive transformations with $N = M$, and identify the transformation with the transformation matrix $\mathbf{T} \in \mathbb{R}^{N\times N}$. We call the transformed signal $\mathbf{y} \in \mathbb{R}^N$ the *coefficients* of $\mathbf{f}$ regarding $\mathbf{T}$. Each entry in $\mathbf{y}$ is a superposition of the elements of $\mathbf{f}$, weighted by the transformation kernels $\mathbf{T}(k,l)$

$$\mathbf{y}(k) = \sum_{l=1}^{N} \mathbf{f}(l)\mathbf{T}(k,l)$$

for all $1 \leq k \leq N$. To reconstruct the input signal, the inverse of $\mathbf{T}$ is used as a second transformation matrix. Therefore, we employ orthonormal (unitary in the case of the discrete Fourier transform) transformation matrices, so that $\mathbf{T}^{-1} \in \mathbb{R}^{N\times N}$ exists and is given by $\mathbf{T}^{-1} = \mathbf{T}^T$.

Definition 2.2 *Given a discrete coefficient vector* $\mathbf{y} \in \mathbb{R}^N$ *along with an orthonormal transformation* $\mathbf{T} \in \mathbb{R}^{N\times N}$*, the* inverse transform *is given by*

$$\mathbf{f} = \mathbf{T}^{-1}\mathbf{y} = \mathbf{T}^T\mathbf{y}. \tag{2.3}$$

Applying equations (2.2) and (2.3), we are able to express the input signal in two domains: the original (spatial in the case of images) domain of the signal and a domain defined by the transformation matrix. Since $\mathbf{T}$ is orthonormal, its rows form an orthonormal system. If we regard $\mathbf{T}^T$ columnwise, i.e. $\mathbf{T}^T = (\mathbf{t}_1 \mid \cdots \mid \mathbf{t}_N)$, the inverse transform is a superposition of these columnvectors:

$$\mathbf{f} = \mathbf{T}^T\mathbf{y} = \sum_{k=1}^{N} \mathbf{y}(k)\mathbf{t}_k.$$

Hence, the system $\{\mathbf{t}_k\}_{1\leq k\leq N}$ forms a basis of the original domain. The transformation coefficients $\mathbf{y}(k)$ act as weights of each basis vector.

Note that due to the choice of an orthonormal transformation matrix $\mathbf{T}$, the energy of the signal remains unchanged

$$\|\mathbf{f}\|^2 = \mathbf{f}^T\mathbf{f} = \mathbf{f}^T\mathbf{T}^T\mathbf{T}\mathbf{f} = \mathbf{y}^T\mathbf{y} = \|\mathbf{y}\|^2. \tag{2.4}$$

Thus, the transformation itself is lossless and does not compress the data. It is instead used to compact the energy and enable better *quantization*.

Quantization

In general, a quantizer reduces the accuracy of a large continuous set, see [40]. It maps input values from an interval $\mathcal{R} \subseteq \mathbb{R}$, known as the *quantization support*, to a discrete set $\mathcal{L}$.

Definition 2.3 *A* quantizer $q : \mathcal{R} \to \mathcal{L}$ *is defined by a set of pairwise disjoint* cells $\mathcal{C} = \{C_i, i \in \mathcal{I}\}$, *where* $\mathcal{R} = \bigcup_{i\in\mathcal{I}} C_i$, *and a set of* levels $\mathcal{L} = \{y_i, i \in \mathcal{I}\}$, *where* $\mathcal{I} = \{1, \dots, K\}$ *is an index set. It is given by*

$$q(x) = \sum_{i=1}^{K} y_i \mathbb{1}_{C_i}(x), \tag{2.5}$$

where $\mathbb{1}_{C_i}(x) = 1$ *if* $x \in C_i$ *and* $\mathbb{1}_{C_i}(x) = 0$ *otherwise.*

A value $x \in \mathcal{R}$ is thus quantized to the level y_i of its corresponding cell C_i. It is represented by its index $i \in \mathcal{I}$, which is efficiently encoded. Each cell is given by $C_i = (a_{i-1}, a_i]$, with the a_i's acting as thresholds. If the levels y_i are equispaced and the threshholds midway between them, the quantizer is said to be *uniform*, and the *quantization step size* is denoted by Δ. An example of a uniform quantizer can be seen in figure 2.2.

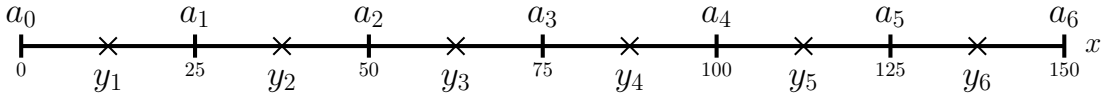


Figure 2.2: An uniform quantizer that quantizes $\mathcal{R} = [0, 150]$ to six levels. The quantization step size is $\Delta = 25$.

The quantization step following the transformation is why the choice of transformation matrix, and thus transform, is critical for a good performance of a compression scheme. The aim of transforming a signal $\mathbf{f}$ is to decorrelate it and compact the contained information to few significant coefficients in $\mathbf{y}$. Such a decorrelated signal has a few large coefficients, while most coefficients are of small magnitude and irrelevant. During quantization the irrelevant coefficients are set to 0, and only large, significant coefficients are encoded.

The Discrete Fourier Transform

One of the fundamental transforms in signal processing is the *discrete Fourier transform* (DFT). Even though there are no direct applications in image processing, it demonstrates important intuitions. We give a very brief overview, focusing on those intuitions. For a more detailed, technical analysis we refer to [67].

The fundamental idea of the Fourier transform is to represent a signal as a weighted combination of *elemental frequencies*. It represents the signal equivalently in the spatial and *frequency domain*.

Definition 2.4 *The one-dimensional* discrete Fourier transform *is defined by the transformation matrix* $\mathbf{B} \in \mathbb{R}^{N\times N}$, *given by*

$$[\mathbf{B}]_{k,l} = \frac{1}{\sqrt{N}} \mathrm{e}^{-\mathrm{i}2\pi \frac{(k-1)(l-1)}{N}} \tag{2.6}$$

for all $1 \le k, l \le N$.

We note that the DFT is usually defined for $0 \leq k, l \leq N-1$ and with a factor $\frac{1}{N}$. The representation in 2.6 was chosen to comply with the transformation framework introduced above. The matrix $\mathbf{B}$ is complex, so the inverse DFT is given by the conjugate transpose $\mathbf{B}^*$. For a discrete signal $\mathbf{f} \in \mathbb{R}^N$, the complete *Fourier spectrum* is given by

$$\hat{\mathbf{f}} \equiv \mathbf{y} = \mathbf{B}\mathbf{f},$$

and the k^{th} *Fourier coefficient* by

$$\hat{\mathbf{f}}(k) \equiv \mathbf{y}(k) = \sum_{l=1}^{N} \mathbf{f}(l) \frac{1}{\sqrt{N}} \mathrm{e}^{-\mathrm{i}2\pi \frac{(k-1)(l-1)}{N}}$$

for all $1 \leq k \leq N$. Following our discussion above, the DFT decomposes $\mathbf{f}$ into its constituting frequencies, given by the vectors

$$\mathbf{b}_k = \frac{1}{\sqrt{N}} \left(\mathrm{e}^{-\mathrm{i}2\pi \frac{(k-1)\cdot 0}{N}}, \ldots, \mathrm{e}^{-\mathrm{i}2\pi \frac{(k-1)\cdot (N-1)}{N}} \right)^T$$

for all $1 \leq k \leq N$. Each Fourier coefficient $\hat{\mathbf{f}}(k)$ weighs the amount of its corresponding elemental frequency $\mathbf{b}_k$ present in $\mathbf{f}$. Lower values of k correspond to low spatial frequencies $\mathbf{b}_k$, i.e. vectors that are smooth and slowly oscillating. In particular, for $k = 0$ we have $\mathbf{b}_0 = \frac{1}{\sqrt{N}}(1, \ldots, 1)^T$, a constant vector. As k increases, so does the frequency, leading to faster oscillating vectors $\mathbf{b}_k$.

Due to the complex quantity in the frequency domain, the DFT is cumbersome in practical image compression applications and thus rarely used directly. But thanks to the *Fast Fourier Transform* the discrete Fourier transform can be computed efficiently in $\mathcal{O}(n \log n)$, see [11]. As other transforms (such as the discrete cosine transform, see chapter 2.3) are computed efficiently by an adapted Fast Fourier Transform as in [61], it is still a very important algorithm in signal processing as a whole.

Image Evaluation

To evaluate the performance of coding schemes in lossy image compression, *rate-distortion* theory is used. It analyzes the trade-off between the coding rate R and induced distortion D. Specifically, a coding scheme aims to minimize

$$\min D + \lambda R,$$

where $\lambda \in \mathbb{R}^+$ is a weighing parameter. The rate is usually understood as the number of bits to be stored. For the notion of distortion, no uniformly accepted measure has been developed so far.

We introduce two *full reference metrics* that compare the approximation $\widetilde{\mathbf{I}}$ with a known reference image $\mathbf{I}$. The standard objective quantative used in classic compression is the *peak signal-to-noise ration* (PSNR). We first measure the absolute distortion between $\mathbf{I}$ and $\widetilde{\mathbf{I}}$ as the average squared intensity differences per pixel, given by the *mean square error* (MSE)

$$\bar{\eta}^2(\widetilde{\mathbf{I}}, \mathbf{I}) = \frac{\eta^2(\widetilde{\mathbf{I}}, \mathbf{I})}{|\mathrm{P}|},$$

where

$$\eta(\widetilde{\mathbf{I}}, \mathbf{I}) = \sqrt{\sum_{p \in \mathrm{P}} \left| \widetilde{\mathbf{I}}(p) - \mathbf{I}(p) \right|^2} \tag{2.7}$$

denotes the approximation error. To put the distortion in perspective to the parameters of the original image, the PSNR defines a logarithmic rescaling of the MSE.

Definition 2.5 *Given an image* $\mathbf{I}$ *along with an approximation* $\widetilde{\mathbf{I}}$*, the peak signal-to-noise ration is defined as*

$$\mathrm{PSNR}(\widetilde{\mathbf{I}}, \mathbf{I}) := 10 \cdot \log_{10} \left(\frac{(2^r - 1)^2}{\bar{\eta}^2(\widetilde{\mathbf{I}}, \mathbf{I})} \right), \tag{2.8}$$

where r *is the bit length of* $\mathbf{I}$*.*

The PSNR is the ratio between the maximum possible power of the image and the corrupting noise, which is the error introduced by the compression. It is measured in decibel (dB) and since it is the reciprocal of the mean square error, it increases as the approximation quality improves and the mean square error decreases.

The PSNR is easy to evaluate and has a clear physical meaning and is thus very popular. But as it simply compares the intensity values on single pixels and does not put them in context to their neighbourhood, it fails to accurately model human perception. In [96] images are presented that have the same PSNR but vary in perceived visual quality. Examples of such images are shown in Figure 2.3. It can clearly be seen that some distortions affect the human visual perception much more than others. For example, the distortion introduced by filtering with a Gaussian is much more prominent than mean-shifting the image.

To model human perception, the *structural similarity index measure* (SSIM) was introduced in [96]. It relies on the assumption that the human visual system is highly adapted to perceiving structural information, while also considering luminance and contrast masking terms. Opposed to the absolute errors measured by the PSNR, the SSIM takes strong inter-dependencies among pixels into account and estimates *perceived errors*.

Given an original image $\mathbf{I}$ and an approximation $\widetilde{\mathbf{I}}$, the SSIM is composed of three weighted components, comparing the luminance $l(\mathbf{I}, \widetilde{\mathbf{I}})$, contrast $c(\mathbf{I}, \widetilde{\mathbf{I}})$ and structure $s(\mathbf{I}, \widetilde{\mathbf{I}})$. The following comparison functions were introduced in [96] and are still widely used today.

The luminance similarity is a function of the estimated mean intensities $\mu_{\mathbf{I}}$ and $\mu_{\widetilde{\mathbf{I}}}$,

$$l(\mathbf{I}, \widetilde{\mathbf{I}}) = \frac{2\mu_{\mathbf{I}}\mu_{\widetilde{\mathbf{I}}} + C_1}{\mu_{\mathbf{I}}^2 + \mu_{\widetilde{\mathbf{I}}}^2 + C_1}, \tag{2.9}$$

(a) Original image. SSIM = 1.

(b) Mean-shifted image. SSIM = 0.9899.

(c) Salt and pepper noise. SSIM = 0.7548.

(d) Gaussian filtered. SSIM = 0.7321.

Figure 2.3: Comparison of images with different types of distortion, all with a PSNR of approximately 24.609. The parameters for each distortion have been fine tuned to result in an equal PSNR.

where $C_1 \in \mathbb{R}^+$ is a constant chosen to avoid instability. The contrast similarity is given as a function of the estimated standard deviations $\sigma_{\mathbf{I}}$ and $\sigma_{\widetilde{\mathbf{I}}}$

$$c(\mathbf{I}, \widetilde{\mathbf{I}}) = \frac{2\sigma_{\mathbf{I}}\sigma_{\widetilde{\mathbf{I}}} + C_2}{\sigma_{\mathbf{I}}^2 + \sigma_{\widetilde{\mathbf{I}}}^2 + C_2}, \tag{2.10}$$

where $C_2 \in \mathbb{R}^+$ is a constant chosen to avoid instability. Lastly, the structure similarity is given as a function of the estimated correlation coefficient $\sigma_{\mathbf{I},\widetilde{\mathbf{I}}}$

$$s(\mathbf{I}, \widetilde{\mathbf{I}}) = \frac{\sigma_{\mathbf{I},\widetilde{\mathbf{I}}} + C_3}{\sigma_{\mathbf{I}}\sigma_{\widetilde{\mathbf{I}}} + C_3}, \tag{2.11}$$

again with a constant $C_3 \in \mathbb{R}^+$ to avoid instability. Combining (2.9), (2.10) and (2.11) results in the SSIM measure

$$\mathrm{SSIM}(\mathbf{I}, \widetilde{\mathbf{I}}) = \left(l(\mathbf{I}, \widetilde{\mathbf{I}})\right)^{\alpha} \left(c(\mathbf{I}, \widetilde{\mathbf{I}})\right)^{\beta} \left(s(\mathbf{I}, \widetilde{\mathbf{I}})\right)^{\gamma}, \tag{2.12}$$

where $\alpha, \beta, \gamma > 0$ adjust the relative importance of each component. The SSIM index maps $\left(\mathbf{I}, \widetilde{\mathbf{I}}\right)$

to $[0, 1]$. Smaller values indicate poorer quality, while $\mathrm{SSIM}(\mathbf{I}, \widetilde{\mathbf{I}}) = 1$ iff $\mathbf{I} = \widetilde{\mathbf{I}}$.

In order to simplify (2.12), the standard choices are $\alpha = \beta = \gamma = 1$ and $C_3 = {}^{C_2}/_2$. This results in the standard SSIM index which we use for the rest of this thesis.

Definition 2.6 *The* standard SSIM index *of an image* $\mathbf{I}$ *and its approximation* $\widetilde{\mathbf{I}}$ *is given by*

$$\mathrm{SSIM}(\mathbf{I}, \widetilde{\mathbf{I}}) := \frac{\left(2\mu_{\mathbf{I}}\mu_{\widetilde{\mathbf{I}}} + C_1\right)\left(2\sigma_{\mathbf{I},\widetilde{\mathbf{I}}} + C_2\right)}{\left(\mu_{\mathbf{I}}^2 + \mu_{\widetilde{\mathbf{I}}}^2 + C_1\right)\left(\sigma_{\mathbf{I}}^2 + \sigma_{\widetilde{\mathbf{I}}}^2 + C_2\right)}. \tag{2.13}$$

We adapt the selection for C_1 and C_2 as in [96], i.e. $C_1 = (0.01L)^2$ and $C_2 = (0.03L)^2$, where $L = 2^r - 1$ is the dynamic range of the pixel values. For 8-bit grayscale images this results in $L = 255$.

2.2 Karhunen-Loève Transform

The Karhunen-Loève transform (KLT) was introduced by Karhunen [55] and Loève [60] as a series expansion for continuous random processes. It is the optimal transform for decorrelating components of a vector following a known correlation model, see [52]. Earlier, Hotelling [43] developed the method of principal componenents, which removes the correlation from the discrete elements of a random variable. As such, the KLT is also known as Hotelling transform and closely related to principal component analysis.

The KLT is a non-separable transformation. Hence, we reorder the luminances from a rectangular grid to a line vector. In particular, we express $\mathbf{I} \in \{0, 1, \ldots, 2^r - 1\}^{X_1 \times X_2}$ as a line vector $\mathbf{f} \in \{0, 1, \ldots, 2^r - 1\}^{X_1 \cdot X_2}$ via

$$\mathbf{f}((x_1 - 1) \cdot X_2 + x_2) = \mathbf{I}(x_1, x_2)$$

for $1 \leq x_1 \leq X_1$ and $1 \leq x_2 \leq X_2$. Therefore, we consider an image as a one-dimensional discrete signal $\mathbf{f} \in \mathbb{R}^N$ in the following discussion. Let $\mathbf{f}$ be a given random signal

$$\mathbf{f} = (f_1, \ldots, f_N)^T \in \mathbb{R}^N,$$

i.e. as a sequence of random variables f_k for $1 \leq k \leq N$. The mean of $\mathbf{f}$ is given by

$$\boldsymbol{\mu}_{\mathbf{f}} = (\mu(f_1), \ldots, \mu(f_N))^T := \mathrm{E}\left[\mathbf{f}\right],$$

where μ is the expected value of a random variable. For simplicity, we consider $\mathbf{f}' = \mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}}$ and assume, without loss of generality, that $\boldsymbol{\mu}_{\mathbf{f}} = 0$. In order to apply the KLT the correlation of the signal is given via its *covariance matrix*.

Definition 2.7 *Given a random vector* $\mathbf{f} \in \mathbb{R}^N$*, its* covariance matrix $\boldsymbol{\Sigma}_{\mathbf{f}} \in \mathbb{R}^{N \times N}$ *is defined by the pairwise covariance between the elements*

$$\boldsymbol{\Sigma}_{\mathbf{f}} := \mathrm{E}\left[(\mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}})(\mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}})^T\right] = \mathrm{E}\left[\mathbf{f}\mathbf{f}^T\right].$$

Since $\mathbf{f}$ is real-valued, its covariance matrix is real and symmetric, i.e., $\mathbf{\Sigma}_\mathbf{f}^T = \mathbf{\Sigma}_\mathbf{f}$. Therefore, $\mathbf{\Sigma}_\mathbf{f}$ is diagonizable via

$$\mathbf{\Sigma}_\mathbf{f} = \mathbf{U}_\mathbf{f}\mathbf{\Lambda}_\mathbf{f}\mathbf{U}_\mathbf{f}^T,$$

where $\mathbf{\Lambda}_\mathbf{f} = \operatorname{diag}(\lambda_1, \ldots, \lambda_N)$ is a real, diagonal matrix containing the eigenvalues of $\mathbf{\Sigma}_\mathbf{f}$ and $\mathbf{U}_\mathbf{f} = (\mathbf{u}_1 | \ldots | \mathbf{u}_N)$ a real orthonormal matrix whose columns are the corresponding eigenvectors of $\mathbf{\Sigma}_\mathbf{f}$. Thus, $\mathbf{\Sigma}_\mathbf{f}\mathbf{u}_k = \lambda_k \mathbf{u}_k$ for all $1 \leq k \leq N$. These eigenvectors form the basis of the KLT.

Definition 2.8 *Given a random signal $\mathbf{f} \in \mathbb{R}^N$ along with its covariance matrix $\mathbf{\Sigma}_\mathbf{f} = \mathbf{U}_\mathbf{f}\mathbf{\Lambda}_\mathbf{f}\mathbf{U}_\mathbf{f}^T$, the Karhunen-Loève transform is defined as*

$$\mathbf{y} = \mathbf{U}_\mathbf{f}^T \mathbf{f}.$$

Consequently, the eigenvectors of $\mathbf{\Sigma}_\mathbf{f}$ are the basis vectors of the KLT, as each component $\mathbf{y}(k)$ is the projection of $\mathbf{f}$ onto $\mathbf{u}_k$, i.e. $\mathbf{y}(k) = \langle \mathbf{u}_k, \mathbf{f} \rangle = \mathbf{u}_k^T \mathbf{f}$ for all $1 \leq k \leq N$, where $\langle \cdot, \cdot \rangle$ denotes the standard inner product. This directly implies the optimal decorrelation property. Let $\mathbf{\Sigma}_\mathbf{y}$ be the covariance matrix of the coefficient vector $\mathbf{y}$. By using the identity $\boldsymbol{\mu}_\mathbf{y} = \mathbf{U}_\mathbf{f}^T \boldsymbol{\mu}_\mathbf{f}$ we obtain

$$\begin{aligned}\mathbf{\Sigma}_\mathbf{y} &= \mathrm{E}\left[\mathbf{y}\mathbf{y}^T\right] = \mathrm{E}\left[\mathbf{U}_\mathbf{f}^T \mathbf{f} (\mathbf{U}_\mathbf{f}^T \mathbf{f})^T\right] \\ &= \mathbf{U}_\mathbf{f}^T \mathrm{E}\left[\mathbf{f}\mathbf{f}^T\right] \mathbf{U}_\mathbf{f} = \mathbf{U}_\mathbf{f}^T \mathbf{\Sigma}_\mathbf{f} \mathbf{U}_\mathbf{f} = \mathbf{\Lambda}_\mathbf{f}.\end{aligned}$$

The covariance matrix of $\mathbf{y}$ is thus diagonal, so the covariance between two disctinct components $\mathbf{y}(i)$ and $\mathbf{y}(j)$ for all $1 \leq i < j \leq N$ is 0. Therefore, the transformed signal is completely decorrelated.

It should be noted that $\mathbf{U}$ is not unique in regard to this property. There are possibly several orthonormal matrices that result in a completely decorrelated signal. But among all orthonormal transformations the KLT maximally compacts the energy of the signal. While the total energy of a signal remains unchanged by a transformation with an orthonormal matrix as shown in (2.4), the KLT packs the most average energy in $K < N$ samples of coefficients.

Following along the lines of Shannons information measures as in [78], and without going into any detail, the information contained in the first K coefficients of $\mathbf{y} = \mathbf{T}\mathbf{f}$ can be represented by

$$E_K(\mathbf{T}) = \sum_{k=1}^{K} \sigma_k^2,$$

The KLT maximizes this information function, see [52], among all orthonormal transforms $\mathbf{T}$, i.e. $E_K(\mathbf{U}_\mathbf{f}^T) \geq E_K(\mathbf{T})$, for all $K < N$. Note that if $\mathbf{T} = \mathbf{U}_\mathbf{f}^\mathbf{T}$, we have $\sigma_{\mathbf{y}(k)}^2 = \lambda_k$. Therefore, we achieve maximum information in the first K components if we order the eigenvalues in descending order, i.e. $\lambda_1 \geq \cdots \geq \lambda_N$. Algorithm 1 shows an image compression scheme exploiting these properties for an image signal $\mathbf{f} \in \mathbb{R}^N$.

Compression is achieved since $K < N$ and since zeros are encoded efficiently. To reconstruct an approximation $\widetilde{\mathbf{f}} \approx \mathbf{f}$ from $\widetilde{\mathbf{y}}$, we simply apply the inverse KLT. Per our discussion in Section 2.1,

Algorithm 1: KLT of image signal $\mathbf{f}$

Input: image signal $\mathbf{f}$ to be compressed.
Output: efficient coefficient vector $\widetilde{\mathbf{y}}$; description of $\mathbf{\Sigma_f}$.

1 Estimate $\mathbf{\Sigma_f}$.
2 Calculate $\mathbf{\Sigma_f} = \mathbf{U_f \Lambda_f U_f^T}$ with $\lambda_1 \geq \cdots \geq \lambda_N$.
3 Choose $K < N$ based on desired information or coding cost.
4 Transform $\mathbf{f}$ via $\mathbf{y} = \mathbf{U_f^T f}$ and set $\mathbf{y}(k) = 0$ for $K < k \leq N$.
5 Encode resulting $\widetilde{\mathbf{y}}$.

it is simply given by

$$\mathbf{f} \approx \widetilde{\mathbf{f}} = \mathbf{U_f} \widetilde{\mathbf{y}}.$$

We observe that the inverse KLT requires the full transformation matrix $\mathbf{U_f}$ to construct the approximation $\widetilde{\mathbf{f}}$. This results in a large overhead, as a lot of data related to the signal has to be transferred as well. Additionally, the underlying correlation between pixels has to be estimated. Clearly this naive approach is not feasible.

In practice, each block to be encoded is first classified into one of several predetermined stochastic classes. But this leads to more distortion of the reconstructed image. As such, it is important to select a good set of stochastical classes to achieve a good performance, see for example [82].

We remark that even though the vector is statistically completely decorrelated, this does not automatically transfer to optimality in transform coding. While it was shown in [38] that the KLT is optimal for high-rate transform coding of Gaussian vectors, there are cases where the KLT is suboptimal, see [28].

2.3 JPEG

The JPEG compression standard was developed by the *Joint Photographic Experts Group* as a collaboration between CCITT and ISO. It was introduced in 1992, see [89], and is the most used digital image format. The JPEG method relies on splitting the image into small 8×8 blocks and quantizing the *Discrete Cosine Transform* (DCT). Unlike the KLT, it is fixed and independent of the data to be transformed.

Discrete Cosine Transform

The DCT was first described by Ahmed, Natarajan and Rao in 1974, see [2]. Four basic types of DCT exist, DCT-I to DCT-IV, see [73], with the DCT-II being most used, including in the JPEG compression scheme. Each type of DCT transforms the signal from the spatial domain to the frequency domain by representing a signal via different cosine frequencies. They differ in their respective choice of boundary conditions, the DCT-II for example extends an input vector $\mathbf{f} \in \mathbb{R}^N$ evenly around the indices $\frac{1}{2}$ and $n + \frac{1}{2}$.

Definition 2.9 *The one-dimensional DCT-II is defined by the transformation matrix* $\mathbf{C} \in \mathbb{R}^{N\times N}$*, given by*

$$[\mathbf{C}]_{i,j} = \left(\frac{2}{N}\right)^{\frac{1}{2}} c_j \cos\left(\frac{(i-1)(j-\frac{1}{2})\pi}{N}\right)$$

for $1 \leq i,j \leq N$*, where* $c_1 = 1/\sqrt{2}$ *and* $c_j = 1$ *otherwise.*

The factor $(2/N)^{\frac{1}{2}} c_j$ ensures the orthonormality of $\mathbf{C}$, so that it fits into the transformation framework introduced in Section 2.1. This factor is usually omitted, which leads to a specific correspondance with the DFT. Due to the boundary conditions shown above, the DCT-II is, up to a factor, equivalent to the DFT of $4N$ inputs of the extended signal. For further processing the factor is inconsequential, as computational steps such as a following quantization can be chosen accordingly.

Multivariate DCTs are simply a separable tensor product of DCTs along each dimension, see [52]. For a two-dimensional rectangular image $\mathbf{I} \in \{0,\dots,255\}^{M\times N}$, the DCT-coefficients are subsequently given by

$$[\mathbf{Y}]_{\nu\omega} = \frac{2c_\nu c_\omega}{\sqrt{MN}} \sum_{i=1}^{M}\sum_{j=1}^{N} [\mathbf{I}]_{i,j} \cos\left(\frac{(\nu-1)((i-\frac{1}{2})\pi}{M}\right) \cos\left(\frac{(\omega-1)(j-\frac{1}{2})\pi}{N}\right)$$

for $1 \leq \nu \leq M$ and $1 \leq \omega \leq N$. Thanks to this separable product, the two-dimensional DCT can be efficiently computed as a combination of two one-dimensional DCTs. Additionally, the DFT of an even function results in the DCT. Consequently, the DCT can be computed very efficiently via the fast Fourier transform with a computational complexity of $\mathcal{O}(N \log N)$ for a signal with N components [73].

JPEG Compression

At the input to the encoder, the source image $\mathbf{I}$ is split into 8×8 blocks and shifted from unsigned integers to signed integers, i.e., from $[0,\dots,2^p-1]^{8\times 8}$ to $[-2^{p-1},\dots,2^{p-1}-1]^{8\times 8}$. Afterwards, the two-dimensional DCT is applied to the shifted image blocks. This results in 64 transform coefficients, the AC coefficient corresponding to the constant basis image and 63 DC coefficients. Each coefficient represents the relative amount of its associated spatial frequency present in the input signal. All 64 spatial frequencies are displayed in Figure 2.4.
As in equation (2.4), the energy of the transformed signal remains unchanged. Compression, and thus distortion, is introduced in the next step: Quantization. Each of the 64 coefficients is quantized to an index by dividing each DCT coefficient by its corresponding quantizer step size and rounding to the nearest integer

$$[\mathbf{Y}_{\text{ind}}]_{i,j} = \text{Round}\left(\frac{[\mathbf{Y}]_{i,j}}{[\mathbf{Q}]_{i,j}}\right) \tag{2.14}$$

for $1 \leq i,j \leq 8$. The step sizes are stored in a Quantization Table $\mathbf{Q} \in \mathbb{R}^{8\times 8}$, which has to be specified by the application at the encoder and controls the compression. When the intention is to compress the image as much as possible, large step sizes are chosen. This leads to low indices after

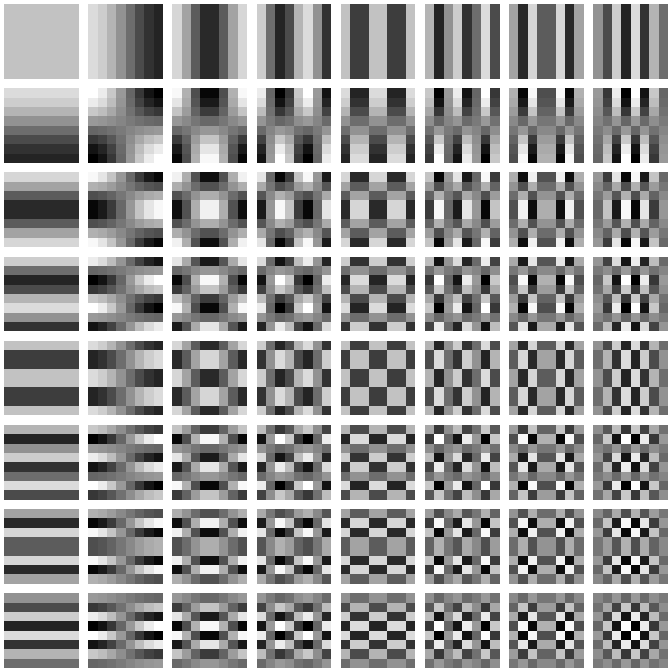


Figure 2.4: Two-dimensional basis images of the DCT. The frequency increases towards the lower right corner.

the division in (2.14), and rounding to the nearest integer will eliminate a substantial amount of coefficients. Ideally, this should be balanced so that visible artifacts are largely supressed. To this end, psychovisual experiments have been performed to determine perceptual thresholds on the concept of just noticeable differences for the visual contribution of each cosine basis function. The result of these experiments led to the standard Quantization Table for JPEG compression

$$\mathbf{Q}_S = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}. \tag{2.15}$$

Note that $\mathbf{Q}_S$ is designed to contain low values in the top left, corresponding to low frequencies, while the quantization step size increases along with the frequency. This is caused by the fact that the human perception is sensitive to small variations in low frequencies, whereas the reconstruction of high frequencies is not as important. Simply put, small perturbations of smooth regions are more noticable than even severe deviations in a high contrast region of the image. For more information about the construction of quantization matrices we refer to [3].

From the standard quantization matrix in (2.15) we derive different quantization levels, see [1],

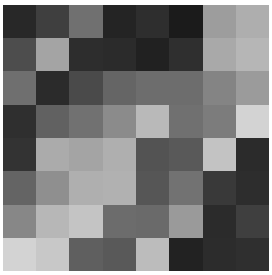
via a quality factor $k \in [1, 100]$ by

$$\mathbf{Q}_k = \begin{cases} \left(\frac{100-k}{50}\right) \mathbf{Q}_S & k > 50 \\ \left(\frac{50}{k}\right) \mathbf{Q}_S & k \leq 50, \end{cases}$$

where $\mathbf{Q}_{50} \equiv \mathbf{Q}_S$. A large quality factor leads to lower values of the quantization matrix and thus less compression and higher quality. For a low quality factor this is reversed.

The resulting sparse matrix $\mathbf{Y}_{\text{ind}}$ comprising the quantized coefficients is then efficiently encoded, e.g. via Huffman coding. Reconstruction of the image reverses the preceding steps. First, the coefficients are reconstructed by multiplying them element-wise with the quantization matrix obtained by the stored quality factor. Afterwards, the inverse DCT is applied and the resulting signal shifted back to its original range. This gives the reconstructed image $\widetilde{\mathbf{I}} \approx \mathbf{I}$.

Figure 2.5 shows the JPEG compression applied to a block of the famous Lena image with different quality factors.



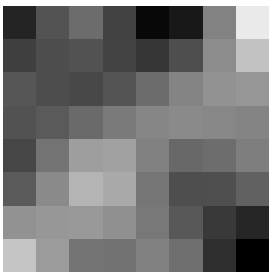
(a) original image block

40	63	112	36	46	27	157	174
77	164	45	43	33	47	169	182
111	43	74	101	109	109	132	155
47	97	113	139	185	112	124	211
51	171	164	175	83	89	195	43
100	143	175	177	86	114	57	45
135	183	196	108	106	154	43	63
211	200	95	89	188	34	44	47

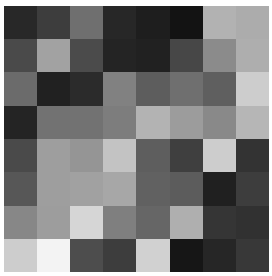
(b) original image values

−143.9	24.8	20	−79.8	−31.1	−24.7	−53.9	17.6
−81.6	−236.5	127.1	−69	5.3	38.9	−26.4	18.9
−70.1	72.1	109	31	17.9	−49	1.2	42.1
−24	28.4	−42.9	−78.3	−109.7	57.8	13.1	−26.5
10.4	−25.9	−17	−22.6	23.1	−7.1	−61.6	58.6
20.6	−52.6	−42.9	−11.8	31.2	−33.6	106.3	14.8
−32.3	12.1	−34.3	23.1	63.1	38.2	30.3	18.5
−5.5	14.3	−10.1	17.2	−55.6	59.5	3.2	9.9

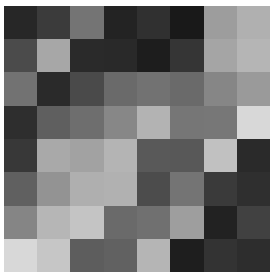
(c) original image coefficients after shift and DCT



(d) Reconstruction with $k = 10$.



(e) Reconstruction with $k = 50$.



(f) Reconstruction with $k = 90$.

Figure 2.5: JPEG compression of a single block with different quality factors

2.4 JPEG 2000

The JPEG compression standard based on the DCT has many advantages and is still the most widely used image compression method, but there are several drawbacks. While the small block size of 8×8 reduces computation time, it leads to blocking artifacts, where neighbouring blocks eliminate a different set of frequencies, which results in different representations of the same structure. Additionally, the codestream is inflexible, multiple resolutions for example are not supported.

To adress these problems and several technical issues, the JPEG committee started creating the *JPEG2000* compression standard and introduced it in December of 2000, see [9]. The JPEG2000 still image coding system is based on a *discrete wavelet transform* (DWT), which enables a multiresolution representation of images.

Wavelets

To get an insight into the working of the JPEG2000 standard, we give a brief overview of *wavelets* and *multi-resolution analyis* on the Hilbert space of square integrable functions on the real line, $\mathrm{L}^2(\mathbb{R})$. For more details we refer to [85]. The aim is to define an orthonormal basis generated by a single function by translation and dilation. To this end, a multi-resolution analysis is defined.

Definition 2.10 *A* multi-resolution analysis *is defined as a sequence of nested resolution spaces* $\mathcal{U}^{(k)} \subset \mathcal{U}^{(k-1)}$ *for* $k \in \mathbb{Z}$*, satisfying the following conditions:*

MR1: *The union of all resolution spaces is dense in* $\mathrm{L}^2(\mathbb{R})$*, i.e.*

$$\bigcup_{k\in\mathbb{Z}} \mathcal{U}^{(k)} = \mathrm{L}^2(\mathbb{R}).$$

MR2: *The intersection of all resolution spaces contains only the zero element, i.e.*

$$\bigcap_{k\in\mathbb{Z}} \mathcal{U}^{(k)} = \{0\}.$$

MR3: *Dilating a signal* $\mathbf{x} \in \mathcal{U}^{(0)}$ *by the factor* 2^k *yields a signal in* $\mathcal{U}^{(k)}$*, i.e.*

$$\mathbf{x}(t) \in \mathcal{U}^{(0)} \Leftrightarrow \mathbf{x}(2^{-k}t) \in \mathcal{U}^{(k)}.$$

MR4: *Translating a signal* $\mathbf{x} \in \mathcal{U}^{(k)}$ *by an integer multiple of* 2^k *does not alter its resolution, i.e.*

$$\mathbf{x}(t) \in \mathcal{U}^{(k)} \Leftrightarrow \mathbf{x}(t - 2^k n) \in \mathcal{U}^{(k)} \text{ for all } n \in \mathbb{Z}.$$

MR5: *There is an orthonormal basis* $\{\varphi_j\}_{j\in\mathbb{Z}}$ *for* $\mathcal{U}^{(0)}$*, so that* φ_j *is a translation of a* scaling function φ*, i.e.*

$$\varphi_j(t) = \varphi(t - j). \tag{2.16}$$

Due to properties **MR3** and **MR4**, an orthonormal basis $\{\varphi_j^k\}_{j\in\mathbb{Z}}$ for $\mathcal{U}^{(k)}$ can be derived via

$$\varphi_j^{(k)}(t) = \sqrt{2^{-k}}\varphi(2^{-k}t - j)$$

Given a scaling function $\varphi \in \mathrm{L}^2(\mathbb{R})$, which characterizes a multi-resolution analysis, it is possible to construct an orthonormal wavelet basis. We denote the orthogonal complement of $\mathcal{U}^{(k+1)}$ in $\mathcal{U}^{(k)}$ as $\mathcal{W}^{(k+1)}$, i.e.

$$\mathcal{W}^{(k+1)} \perp \mathcal{U}^{(k+1)} \text{ and } \mathcal{W}^{(k+1)} \oplus \mathcal{U}^{(k+1)} = \mathcal{U}^{(k)}.$$

In this way, the resolution space $\mathcal{U}^{(k)}$ is decomposed into a smooth space $\mathcal{U}^{(k+1)}$ containing the low frequency functions and a coarse wavelet space $\mathcal{W}^{(k+1)}$ that contains the high frequency functions of $\mathcal{U}^{(k)}$. Since $\mathcal{U}^{(0)} \subset \mathcal{U}^{(-1)}$, $\varphi \in \mathcal{U}^{(0)}$ can be expressed as a linear combination of functions $\varphi_j^{(-1)}$ via

$$\varphi(t) = \sqrt{2}\sum_{j\in\mathbb{Z}} a_j^{(0)}\varphi(2t - j) \tag{2.17}$$

with some coefficients $a_j^{(0)}$ for $j \in \mathbb{Z}$. Treating the coefficients as an element in $\ell^2(\mathbb{R})$ via $a^{(0)} := \left(a_j^{(0)}\right)_{j\in\mathbb{Z}}$, property **MR5** shows that it has unit norm and is orthogonal to all of its 2-translates. This is the condition required for a low-pass synthesis filter, see [85]. Thus, defining a second set of coefficients $a^{(1)} := \left(a_j^{(1)}\right)_{j\in\mathbb{Z}}$ with

$$a_j^{(1)} = (-1)^{j+1}a_{-(j-1)}^{(0)}, \tag{2.18}$$

the 2-translates of $a^{(0)}$ and $a^{(1)}$ together form an orthonormal basis for $\ell^2(\mathbb{Z})$. Finally, let

$$\psi(t) = \sum_{j\in\mathbb{Z}} a_j^{(1)}\varphi(2t - n)$$

for $t \in \mathbb{R}$. The function $\psi \in \mathcal{W}^{(0)}$ is called *mother wavelet*. By defining its translation analogously to (2.16) via $\psi_j(t) = \psi(t - j)$, the collection $\{\psi_j\}_{j\in\mathbb{Z}}$ is an orthonormal basis for $\mathcal{W}^{(0)}$. For every resolution $k \in \mathbb{Z}$, a wavelet in the corresponding orthogonal complement $\mathcal{W}^{(k)}$ is derived by dilating ψ with a factor 2^k: $\psi^{(k)}(t) = \sqrt{2^{-k}}\psi(2^{-k}t)$. The collection of the dilated and translated wavelets $\{\psi_j^{(k)}\}_{j\in\mathbb{Z}}$ form an orthonormal basis for $\mathcal{W}^{(k)}$. Together, $\{\psi_j^{(k)}\}_{j,k\in\mathbb{Z}}$ form an orthonormal wavelet basis of $\mathrm{L}^2(\mathbb{R})$.

We now move to the computation of the discrete wavelet transform. Analogously to the theory above, it gives a multiresolution representation of a discrete signal. At each level the signal is split into its low and high frequency components. Since the regular DWT is rarely applied in practice, we skip its introduction. Instead, it was shown in [62] that the DWT can be implemented as a two-channel subband decomposition of a given discrete signal. Hence, we give a brief overview of it, as it is the transform utilized in image processing.

Given a one-dimensional discrete signal $\mathbf{f} \equiv \mathbf{f}^{(0)} \in \mathbb{R}^N$, the two-channel subband decomposition is given as a series of low-pass and high-pass filters, denoted by h_{low} and h_{high} respectively.

Low-pass samples represent the low-resolution version of $\mathbf{f}$. The low-pass filter is defined by the coefficients of $h_{\text{low}} = a^{(0)}$ as in (2.17) and computed via a convolution, i.e.

$$\mathbf{f}^{(1)}_{\text{low}}(k) = (\mathbf{f}^{(0)} * h_{\text{low}})(k) = \sum_{j\in\mathbb{Z}} \mathbf{f}^{(0)}(j)a^{(0)}(k-j).$$

High-pass samples on the other hand represent the residual version of the original signal. The high-pass filter is analogously defined by the coefficients of $h_{\text{high}} = a^{(1)}$ as in (2.18) and computed via a convolution, i.e.

$$\mathbf{f}^{(1)}_{\text{high}}(k) = (\mathbf{f}^{(0)} * h_{\text{high}})(k) = \sum_{j\in\mathbb{Z}} \mathbf{f}^{(0)}(j)a^{(1)}(k-j).$$

Since half the frequencies in both vectors have been removed, the resulting signals are downsampled by a factor two without any loss of information, which we denote by $\downarrow 2$. We are thus left with two vectors of length $N/2$, with $\mathbf{f}^{(1)}_{\text{low}} \downarrow 2$ comprising the low frequency, and $\mathbf{f}^{(1)}_{\text{high}} \downarrow 2$ high frequency components. This procedure is repeated for the low-pass filtered signals, until a desired resolution is attained. The output coefficients are $\mathbf{f}^{(1)}_{\text{low}} \downarrow 2, \ldots, \mathbf{f}^{(l)}_{\text{low}} \downarrow 2, \mathbf{f}^{(l)}_{\text{high}} \downarrow 2$. Together they are stored in an established order in a coefficient vector $\mathbf{y} \in \mathbb{R}^N$. An example of this process is depicted in figure 2.6.

The original signal is reconstructed by simply reversing the preceding steps. $\mathbf{y}$ is split into individual coefficient vectors which are upsampled by a factor 2 and deconvoluted.

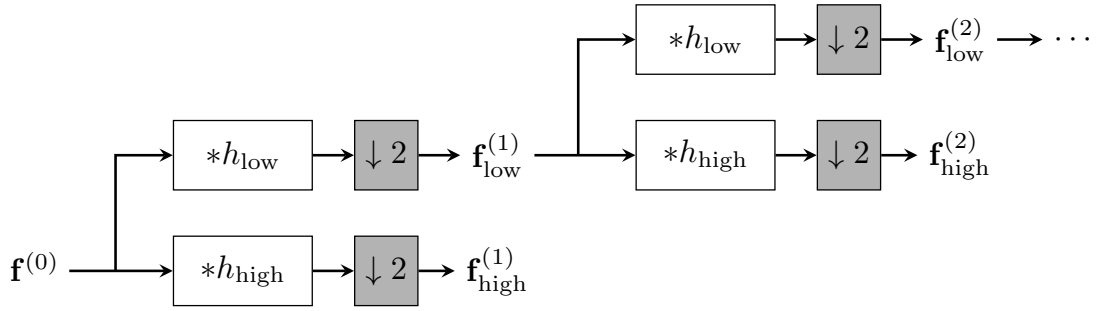
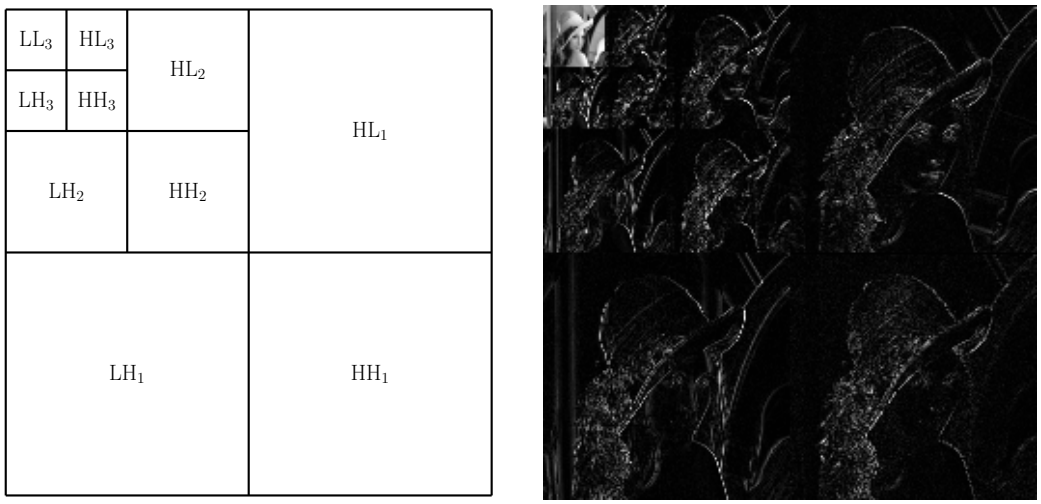


Figure 2.6: Filter bank diagram of the two-channel subband decomposition that outputs the DWT coefficients. Depicted are 2 levels of filtering.

JPEG 2000 compression

At the beginning of the JPEG 2000 compression scheme, the source image $\mathbf{I}$ is split into rectangular blocks. Unlike in regular JPEG compression the blocks can be of any fixed size, even the whole image can be considered as one block.

The two-dimensional DWT is computed by applying the subband transform separably to the columns and then to the rows of each block. Figure 2.7(a) shows a two-dimensional DWT with 3 levels. HH refers to twice high-pass filtered coefficients, HL to coefficients first low-pass filtered and then high-pass filtered and LH to coefficients first high-pass filtered and then low-pass filtered. Smooth components are contained in LL, which is twice low-pass filtered. The subscript denotes



(a) Schematic three level DWT. (b) Three level DWT of Lena

Figure 2.7: A three level DWT given schematically and applied to a natural image.

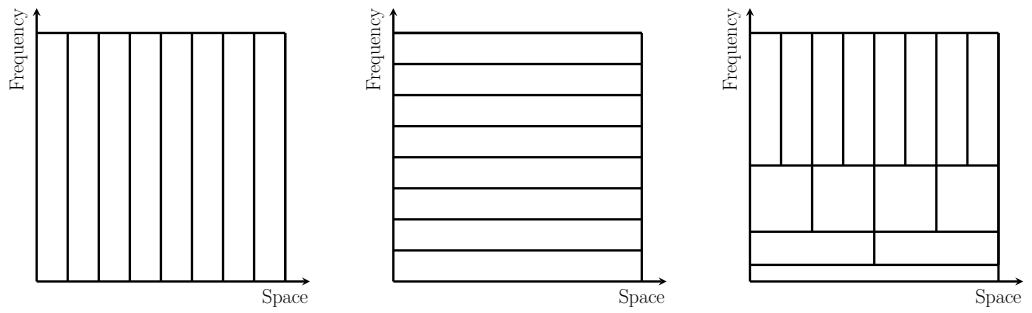
the level of resolution. As above, this is repeated until a desired resolution level is reached. An example of a DWT of Lena is depicted in Figure 2.7(b) with three resolution levels.

After applying the wavelet transform, the resulting coefficient matrix $\mathbf{E}$ is quantized via

$$\mathbf{E}_{\text{ind}}(i,j) = \operatorname{sgn}(\mathbf{E}(i,j)) \left\lfloor \frac{|\mathbf{E}(i,j)|}{\Delta_b} \right\rfloor,$$

where Δ_b depends on the subband b of each coefficient. The resulting sparse matrix $\mathbf{E}_{\text{ind}}$ is then encoded efficiently via EBCOT coding, see [84].

Reconstruction of the image reverses the preceding steps. The coefficients are reconstructed by multiplying them with the corresponding Δ_b. Afterwards, they are upsampled and deconvoluted. This gives the reconstructed image $\widetilde{\mathbf{I}} = \mathbf{I}$.



(a) Signal given in spatial domain. (b) Signal transformed with the DCT. (c) Signal transformed with the DWT.

Figure 2.8: Different representations of a signal.

To explain the advantages of the wavelet based image format, we turn to the resolution of a signal in the spatial and frequency domain, summarized in Figure 2.8. We refer to [35] for a more detailed discussion. In the regular representation, a signal is given as sampled values in the

spatial domain shown in 2.8(a). It is completely localized in space, with no frequency resolution. When transforming the signal with the DCT to the frequency domain, the exact opposite occurs as seen in 2.8(b). While it has a good resolution in the frequency domain, the basis functions of the DCT are periodic and thus have no spatial localization. Transforming a signal with the DWT admits a mixture as seen in 2.8(c). In low frequencies, a high frequency resolution is paired with a low spatial localization. As the frequency increases, so does the spatial resolution, whereas the frequency localization decreases.

This is a result of the dilation and translation of the mother wavelet on each level of the transform. Recall that a wavelet is derived via

$$\psi_j^{(k)}(t) = \sqrt{2^{-k}}\psi(2^{-k}t - j).$$

Therefore, the translation factor j decreases exponentially with the scale. The spatial distance between high frequency wavelets is thus very low, leading to a high spatial resolution. Following the same argumentation, low frequency wavelets are far apart, leading to a low spatial resolution. On the other hand, the dilation factor 2^{-k} also behaves exponentially, which leads to an increased distance in covered frequencies.

This coincides nicely with the human perception of images. For low frequency components of an image, the exact localization is not as important. High frequency components such as edges on the other hand are highly localized. Their exact location is important, while the exact sharpness is not.

Chapter 3

Adaptive Thinning

In the last chapter we have seen how to apply transform coding to image compression. The aim of transform coding is to remove the redundancy in an image by transforming it to a chosen transform domain, e.g. the frequency domain. The key to good compression is to select a target domain that promotes sparsity of the transform coefficients. JPEG and JPEG2000 use representations where the basis images give a certain sense of frequency.

But there are other methods to decorrelate the image without transforming it. One approach is to find a representative, sparse subset of pixels that describe their neighbourhood. By knowing the luminances of these significant pixels, it is ideally possible to recover a good approximation of the original image. Of course special care has to be taken to select a suitable subset of pixels. This may be done via *thinning algorithms* [36], which are schemes that recursively remove pixels according to some specific criterion.

The *Adaptive Thinning* (AT) algorithm by Demaret, Dyn and Iske [15], [21] is such an algorithm. It adopts the concept of a thinning scheme and bases the removal criterion on the location and sampled values of a pixel mask P. At each step of the thinning algorithm, the *significance* of every pixel is evaluated and the least significant pixel is removed. Once a desired amount of significant pixels are attained, the original image is approximated by a linear spline over an anisotropic Delaunay triangulation. We remark that this is not the only scheme utilizing anisotropic triangulations. For an overview of different compression schemes over triangulations we refer to [18].

Like the KLT, AT is an adaptive scheme, i.e. it adapts to the characteristics of a given image. This gives it an inherent advantage over non-adaptive methods such as JPEG or JPEG 2000. Unlike the KLT, it can be efficiently encoded via contextual coding techniques, see [22]

Thinning algorithm

A *thinning* (also known as *decimation* or *simplification*, see [36]) strategy for digital images is a recursive, greedy data removal scheme. In each step, the best candidate for removal, called the *least significant pixel*, is removed from the current set of pixels. It is chosen based on some pre-defined *removal criterion* that depends on the luminance value of each pixel. Recall from chapter 2.1 that all pixels constitute the set P. By recursively removing a pixel in every step we obtain a sequence of nested subsets

$$\mathrm{P}_n \subset \mathrm{P}_{n+1} \subset \cdots \subset \mathrm{P}_N = \mathrm{P}. \tag{3.1}$$

The scheme starts with the full set $\mathrm{P} = \mathrm{P}_N$ consisting of $|\mathrm{P}| = N$ pixels and finishes with a set P_n of $|\mathrm{P}_n| = n$ *most significant pixels*. Here, the amount of significant pixels n is significantly smaller than N. It may be chosen at the start to achieve a desired cardinality of P_n or during the scheme once some predefined condition is met.

After each removal the significance of every pixel has to be re-evaluated. During the iterative process, this re-evaluation is potenitally very expensive computationally. In order to reduce this complexity, the AT algorithm employs special data structures and an advantageous *local error indicator*. Before introducing it, we start by describing the image representation scheme in the next sections. It utilizes linear splines over the Delaunay triangulation of P_n.

3.1 Delaunay Triangulations

A *triangulation* of a set $\mathrm{P}_n \subset \mathbb{R}^2$ containing $n \in \mathbb{N}$ vertices is a set of *triangles* that covers the convex hull of P_n. They are an important tool in numerical mathematics, one example being the construction of computational grids, and are widely used in various algorithms. For a more in depth definition and details concerning the usage of triangulations we refer to [13].

Definition 3.1 *Given a planar point set* P_n *of* n *vertices, a* conformal triangulation *is a decomposition of the convex hull of* P_n *into triangles* $\mathcal{T} \equiv \mathcal{T}(\mathrm{P}_n) = \{T\}_{T\in\mathcal{T}}$*, satisfying the following properties:*

- *the vertex set of* $\mathcal{T}$ *is* P_n*,*
- *the intersection of two distinct triangles is either empty, a vertex or an edge,*
- $\bigcup_{T\in\mathcal{T}} T = \mathrm{Conv}(\mathrm{P}_n)$*.*

A triangle *is in this sense defined as the convex hull of three noncollinear planar points.*

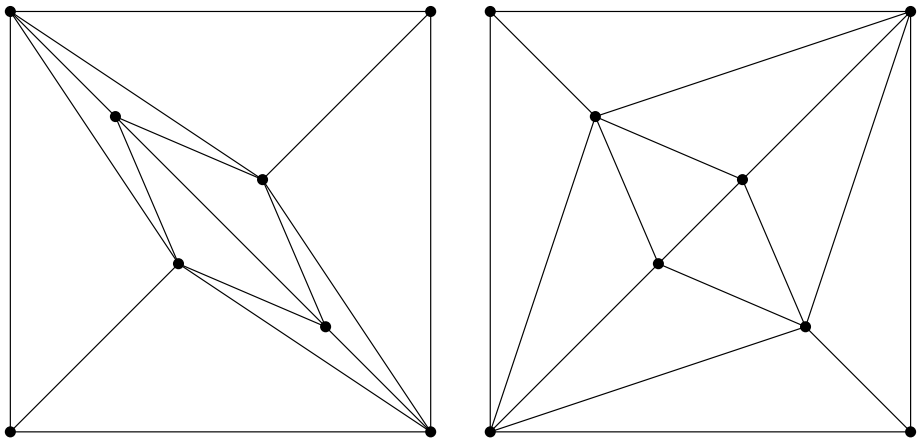
In this chapter the vertices in P_n will correspond to a set of pixels. While a conformal triangulation of a pixel set would already be adequate to approximate an image via linear splines, it is not unique. This would be a severe limitation of the adaptive thinning scheme, as the triangulation would have to be encoded as well. To this end, we introduce *Delaunay triangulations*.

Definition 3.2 *A* Delaunay triangulation $\mathcal{D} \equiv \mathcal{D}(\mathrm{P}_n)$ *of a planar point set* P_n *is a conformal triangulation such that the circumcircle of any triangle* $T \in \mathcal{D}$ *does not contain any point of* P_n *in its interior.*

An example of a conformal and a Delaunay triangulation of the same point set can be seen in Figure 3.1. Note that while the triangulation in 3.1(a) contains long, thin triangles, the Delaunay triangulation over the same point set in 3.1(b) maximizes the minimum angle.

Delaunay triangulations have several important benefits compared to regular conformal triangulations:

- if no four points in P_n are co-circular, i.e. if no 4 points lie on one circle, then the Delaunay triangulation is unique,



(a) Conformal triangulation. (b) Delaunay triangulation.

Figure 3.1: Two different triangulations over a point set consisting of 8 points.

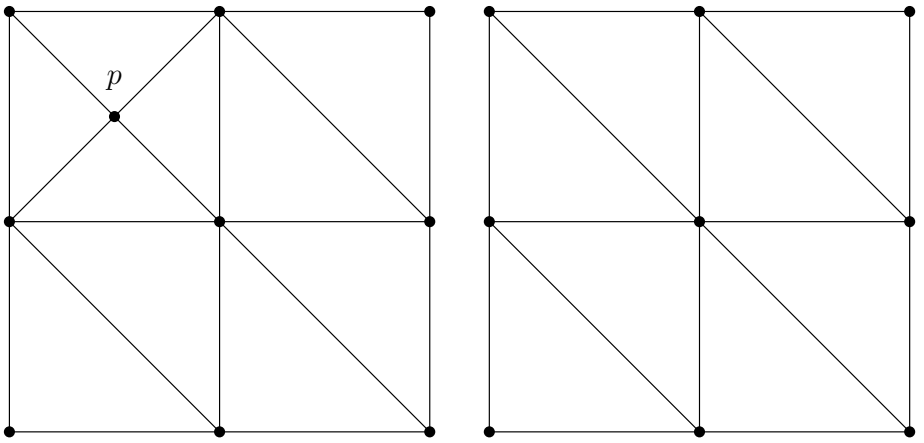
- the Delaunay triangulation of a point set P_n of cardinality $|\mathrm{P}_n| = n$ can be computed efficiently in $\mathcal{O}(n \log(n))$ steps,
- for any vertex $p \in \mathrm{P}_n$, its removal $\mathcal{D}(\mathrm{P}_n \setminus p)$ can be computed by a *local* update.

Of course the first property is not fulfilled for all point sets. In particular, if we consider a point set P consisting of pixels on a regular grid (as is standard for digital images) or a subset thereof, this condition will usually not be fulfilled. Fortunately there exist efficient, reproducible computational methods for choosing a Delaunay triangulation without ambiguity in such a case. An example of this, namely *simulation of simplicity*, can be found in [27]. The basic idea is to perturb the given pixels slightly in a predetermined way, while keeping the perturbation small enough to not change the nondegenerate positions of points relative to each other. Accordingly, we will from here on out assume that the Delaunay triangulation $\mathcal{D}$ of a point set is unique.

Even though the Delaunay triangulation can be computed efficiently in $\mathcal{O}(n \log(n))$ steps, computing it after every step would be too expensive. The pixel removal was therefore designed to only require a *local update*. Figure 3.2 shows the removal of a pixel $p \in \mathrm{P}_n$. In the resulting Delaunay triangulation $\mathcal{D}(\mathrm{P}_n \setminus p)$ only the direct cell (i.e. the domain consisting of all triangles containing p as a pixel) has to be retriangulated. It is consequently computationally very cheap to remove a pixel during the thinning process, as the local update required only concerns few pixels. The Delaunay triangulation in 3.2(b) is only calculated based on four pixels. In Figure 3.2 four triangles in the cell of p in 3.2(a) are replaced by two triangles in 3.2(b) after its removal. All other cells remain unchanged.

3.2 Image representation

Next, we consider bivariate functions $L : \mathrm{Conv}(\mathrm{P}_n) \to \mathbb{R}$ that rely on a triangulation $\mathcal{T}(\mathrm{P}_n)$ of a planar point set P_n. We choose to utilize piecewise linear polynomials over each triangle. In particular, the AT scheme requires continuous, piecewise linear polynomials over a given triangulation.



(a) Delaunay triangulation of a pixel set before pixel removal.

(b) Delaunay triangulation of the pixel set with one pixel removed.

Figure 3.2: Pixel removal of p from a pixel set P_n.

Definition 3.3 *Given a planar point set* P_n *along with a conformal triangulation* $\mathcal{T}(\mathrm{P}_n)$*, the* linear splines space $\mathcal{S}_\mathcal{T}$ *is defined as*

$$\mathcal{S}_\mathcal{T} := \{L \in \mathscr{C}(\mathrm{Conv}(\mathrm{P}_n)) : L|_T \in \Pi_1 \text{ for all } T \text{ in } \mathcal{T}\},$$

where Π_1 *denotes the linear space of all bivariate linear polynomials.*

The linear spline space $\mathcal{S}_\mathcal{T}$ contains all continuous functions over $\mathrm{Conv}(\mathrm{P}_n)$ whose restriction to any triangle $T \in \mathcal{T}$ is a linear polynomial. We call elements of $\mathcal{S}_\mathcal{T}$ simply *linear splines over* $\mathcal{T}(\mathrm{P}_n)$. Piecewise bivariate linear functions have a simple representation and are uniquely defined by the values $L(p)$ at each point $p \in \mathrm{P}_n$. Hence, $\mathcal{S}_\mathcal{T}$ forms a finite dimensional linear function space with its dimension given by the number n of vertices in P_n. An example of a linear spline over the Delaunay triangulation from Figure 3.1(b) is shown in Figure 3.3.

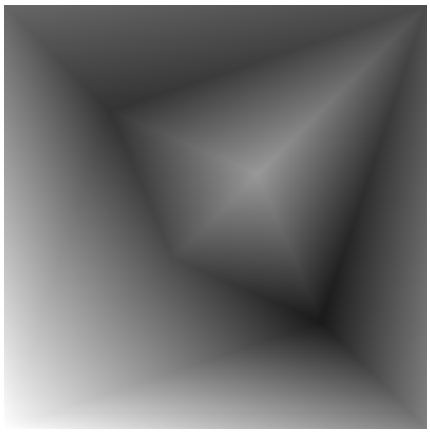


Figure 3.3: Linear spline over the Delauny triangulation from Figure 3.1(b)

AT applied to a digital image $\mathbf{I}$ returns a set of adaptively chosen significant pixels P_n along with their luminances $\mathbf{I}(p)$ for all $p \in \mathrm{P}_n$. We now show how to reconstruct an approximation $\widetilde{\mathbf{I}}$ to $\mathbf{I}$ from this data. To this end, recall from Chapter 2.1 that a digital image $\mathbf{I}$ is given as a

collection of luminances over a set of pixels P. In order to approximate $\mathbf{I}$, we thus have to ensure a pixelwise approximation for every pixel $p \in \mathrm{P}$, i.e.

$$\mathbf{I}(p) \approx \widetilde{\mathbf{I}}(p).$$

The basis for this approximation is the function $L_{\mathbf{I},\mathcal{D}} \in \mathcal{S}_{\mathcal{D}}$ that interpolates $\mathbf{I}$ on P_n. As per our previous considerations, it is uniquely defined by $\mathcal{D} \equiv \mathcal{D}(\mathrm{P}_n)$ and the luminances $\mathbf{I}(p)$ via

$$L_{\mathbf{I},\mathcal{D}}(p) = \mathbf{I}(p)$$

for all significant pixels $p \in \mathrm{P}_n$. An approximation $\widetilde{\mathbf{I}}$ is computed by letting $\widetilde{\mathbf{I}}(p)$ be defined by the rounded value of $L_{\mathbf{I},\mathcal{D}}(p)$. For a digital image the luminances are represented as an r-bit unsigned integer, but the value of $L_{\mathbf{I},\mathcal{D}}$ at each pixel $p \in \mathrm{P}$ will generally not be an integer. We thus have to discretize it, which is done by rounding to the nearest integer.

Definition 3.4 *Given an image* $\mathbf{I}$ *on a pixel set* P *along with a set of significant pixels* $\mathrm{P}_n \subset \mathrm{P}$*, the* adaptive thinning reconstruction $\widetilde{\mathbf{I}}$ *is given by*

$$\widetilde{\mathbf{I}}(p) = \mathrm{Round}\left(L_{\mathbf{I},\mathcal{D}}(p)\right) \approx \mathbf{I}(p) \tag{3.2}$$

for every pixel $p \in \mathrm{P}$*.*

In order for (3.2) to make sense, and thus for $\mathcal{S}_{\mathcal{D}}$ to be an approximation space for the image $\mathbf{I}$, the convex hulls $\mathrm{Conv}(\mathrm{P})$ and $\mathrm{Conv}(\mathrm{P}_n)$ have to coincide. Accordingly, in the initial perturbation of pixel positions, the four corners of $\mathbf{I}$ are unperturbed and the other boundary pixels are perturbed along the edges of $\mathrm{Conv}(\mathrm{P})$. The four corner pixels will not be removed during the thinning process and are thus contained in every set of significant pixels P_n for any n.

3.3 Selection of Significant Pixels

As the reconstruction $\widetilde{\mathbf{I}}$ is strongly dependent on the set of significant pixels P_n, a good choice of P_n is crucial and is discussed in depth in [20]. In particular, the linear spline $L_{\mathbf{I},\mathcal{D}(\mathrm{P}_k)}$ should be close to the original image $\mathbf{I}$ for every subset P_k in the sequence (3.1), with $n \leq k \leq N$. Recall from Chapter 2.1 that the quality of an image approximation can be measured by the peak signal to noise ratio, see (2.8). Since the PSNR is inversely proportional to the MSE, we construct subsets $\mathrm{P}_n \subset \mathrm{P}$ in such a way that the approximation error function η in (2.7) is small. For notational purposes we will denote $\eta(\widetilde{\mathbf{I}}, \mathbf{I}) \equiv \eta(\mathrm{P}_n, \mathrm{P})$ if $\widetilde{\mathbf{I}}$ is constructed as in (3.2) with $\mathcal{D} \equiv \mathcal{D}(\mathrm{P}_n)$.

The *significance* of a pixel is defined via the approximation error. It is the error incurred by its removal from the set P_n.

Definition 3.5 *Given an image* $\mathbf{I}$ *on a pixel set* P *along with a pixel subset* $\mathrm{P}_n \subseteq \mathrm{P}$*, the* significance *of a pixel* $p \in \mathrm{P}_n$ *is defined as*

$$\eta(p) = \eta\left(\mathrm{P}_n \setminus p, \mathrm{P}\right).$$

In [21] it was shown that the AT algorithm is significantly improved by considering *least significant pixel pairs* instead of single pixels. A pair of pixels is said to be least significant in P_n if it minimizes the significance, and thus the approximation error, among all possible pixel pairs.

Definition 3.6 *Given an image* $\mathbf{I}$ *on a pixel set* P *along with a pixel subset* $\mathrm{P}_n \subseteq \mathrm{P}$*, the pixels* $\{p_1^*, p_2^*\} \subset \mathrm{P}_n \subset \mathrm{P}$ *are a* least significant pixel pair *in* P_n *iff*

$$\eta(p_1^*, p_2^*) = \min_{\{p_1,p_2\}\subset \mathrm{P}_n} \eta(p_1, p_2),$$

where the significance of a pixel pair $\{p_1, p_2\} \subset \mathrm{P}_n$ *is simply*

$$\eta(p_1, p_2) = \eta\left(\mathrm{P}_n \setminus \{p_1, p_2\}, \mathrm{P}\right).$$

Finally, a pixel is said to be least significant in P_n if it belongs to a least significant pixel pair and has the smaller significance among the pair.

Definition 3.7 *Given an image* $\mathbf{I}$ *on a pixel set* P *along with a subset* $\mathrm{P}_n \subseteq \mathrm{P}$*, a pixel* $p^* \in \mathrm{P}_n$ *is* least significant *in* P_n *iff it belongs to a least significant pixel pair* $(p^*, p) \subset \mathrm{P}_n \times \mathrm{P}_n$ *and satisfies*

$$\eta(p^*) \leq \eta(p).$$

3.4 Post-processing

Given a certain amount of significant pixels n, we wish to approximate an image $\mathbf{I}$ optimally . To this end, we aim to find P_n^*, the set of pixels that minimizes the approximation error among all sets of pixels with size n

$$\eta(\mathrm{P}_n^*, \mathrm{P}) = \min_{\substack{\mathrm{P}_n\subset\mathrm{P},\\ |\mathrm{P}_n|=n}} \eta(\mathrm{P}_n, \mathrm{P}).$$

Unfortunately, finding such a set is NP-hard and therefore infeasible to solve algorithmically. This is the reason a greedy thinning algorithm is used to approximate the optimal set P_n^*. Even though the AT algorithm chooses the optimal set in each step as seen above, the resulting set of significant pixels P_n may differ substantially from P_n^*. To improve the performance of the scheme, a *pixel exchange* was introduced by Demaret and Iske in [17] as a local optimization post-processing procedure.

Given a set $\mathrm{P}_n \subset \mathrm{P}$, let $\overline{\mathrm{P}_n} = \mathrm{P} \setminus \mathrm{P}_n$ be the complement of P_n in P. A pixel pair $(p_1, \bar{p}_2) \in \mathrm{P}_n \times \overline{\mathrm{P}_n}$ is said to be *exchangeable* iff the removal of p_1 and a subsequent addition of $\bar{p}_2$ to P_n reduces the error incurred by approximation, i.e. iff $\eta((\mathrm{P}_n \setminus \{p_1\}) \cup \{\bar{p}_2\}, \mathrm{P}) < \eta(\mathrm{P}_n, \mathrm{P})$. If there are no further exchangeable pairs of pixels $(p_1, \bar{p}_2) \in \mathrm{P}_n \times \overline{\mathrm{P}_n}$, the Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$ is *locally optimal* and is used from here on out.

Having aquired a locally optimal set of significant pixels, the PSNR is further reduced by a *least squares approximation*, see [48] for a brief introduction. It was first integrated into the AT algorithm in [21].

Given the locally optimal set P_n of significant pixels, there is no guarantee that the linear spline constructed as an interpolant to $\mathbf{I}$ over P_n in (3.2) is the best possible linear spline $L^*_{\mathbf{I},\mathcal{D}} \in \mathcal{S}_\mathcal{D}$.

Definition 3.8 *Given an image* $\mathbf{I}$ *on* P *and a set of most significant pixels* $\mathrm{P}_n \subset \mathrm{P}$*, the linear spline* $L^*_{\mathbf{I},\mathcal{D}} \in \mathcal{S}_\mathcal{D}$ *is called the* best approximation *to* $\mathbf{I}$ *from* $\mathcal{S}_\mathcal{D}$ *iff it minimizes*

$$\sum_{p\in\mathrm{P}} \left| L^*_{\mathbf{I},\mathcal{D}}(p) - \mathbf{I}(p) \right|^2 = \min_{L\in\mathcal{S}_\mathcal{D}} \sum_{p\in\mathrm{P}} |L(p) - \mathbf{I}(p)|^2 . \tag{3.3}$$

Since $\mathcal{S}_\mathcal{D}$ is a finite-dimensional linear space and $\mathrm{P}_n \subset \mathrm{P}$, the best approximation exists and is unique. $\widetilde{\mathbf{I}}$ constructed from $L^*_{\mathbf{I},\mathcal{D}}$ is the reconstruction of $\mathbf{I}$ output by the AT algorithm. It is determined by the set of significant pixels P_n (along with its unique Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$) and its optimal luminances $\mathbf{I}^*(p) = L^*_{\mathbf{I},\mathcal{D}}(p)$ for $p \in \mathrm{P}_n$. The number of parameters describing $L^*_{\mathbf{I},\mathcal{D}}$ depends on the amount n of significant pixels.

3.5 Adaptive Thinning Algorithm

Having discussed all necessary steps, we summarize the AT algorithm in Algorithm 2. We are given an image $\mathbf{I} \in \{0, 1, \ldots, 2^r - 1\}^\mathrm{P}$ for a bit length r as luminances of a set of pixels arrayed in a rectangular grid with resolution $N = X_1 \times X_2$. Additionally, the desired amount n of significant pixels as well as the amount s of post-processing exchanges are specified. n can also be conditionally set during the algorithm once the approximation quality has fallen under a given threshold, but this does not change the algorithm. Each of the possible s post-processing pixel exchanges strictly reduces the approximation error, until a potential locally optimal Delaunay triangulation is attained. For large data sets it is not always feasible to compute the locally optimal Delaunay triangulation, thus an upper bound on pixel exchanges is necessary.

Algorithm 2: Adaptive Thinning

Input: original image $\mathbf{I}$ on original pixel set P_N; amount n of significant pixels; amount s of post-processing swaps

Output: significant pixels P^*_n; quantized luminances $\mathbf{I}^*_\mathrm{q}$.

1 Calculate unique Delaunay triangulation $\mathcal{D}(\mathrm{P}_N)$ after simulation of simplicity.
2 **for** $k = 1, \ldots, N - n$ **do**
3 Find a least significant pixel $p_{N-k+1} \in \mathrm{P}_{N-k+1}$.
4 Let $\mathrm{P}_{N-k} = \mathrm{P}_{N-k+1} \setminus \{p_{N-k+1}\}$.
5 Retriangulate the cell of p_{N-k+1} and update significance of affected pixels.
6 Convert resulting set of pixels P_n into a locally more optimal set P^*_n via s pixel exchanges.
7 Perform least squares approximation which results in optimal luminances $\mathbf{I}^*$ over P^*_n.
8 Encode significant pixels P^*_n and quantized luminances $\mathbf{I}^*_\mathrm{q}$ via contextual coding.

After the set of significant pixels P^*_n is adaptively chosen and quantized luminances $\mathbf{I}^*_q$ are determined, they have to be encoded efficiently. To this end, *contextual coding* was utilized in [22]. Briefly put, contextual coding makes use of causal information to give context to the mode of the

current symbol at the encoder and decoder. Information to be exploited are clusters of significant pixels and the size of edges of triangles contrasted to their difference in luminances.

Finally, the reconstruction of the image at the decoder is shown in Algorithm 3.

Algorithm 3: Reconstruction of $\mathbf{I}$

Input: significant pixels P_n^*; quantized luminances $\mathbf{I}_{\mathrm{q}}^*$.

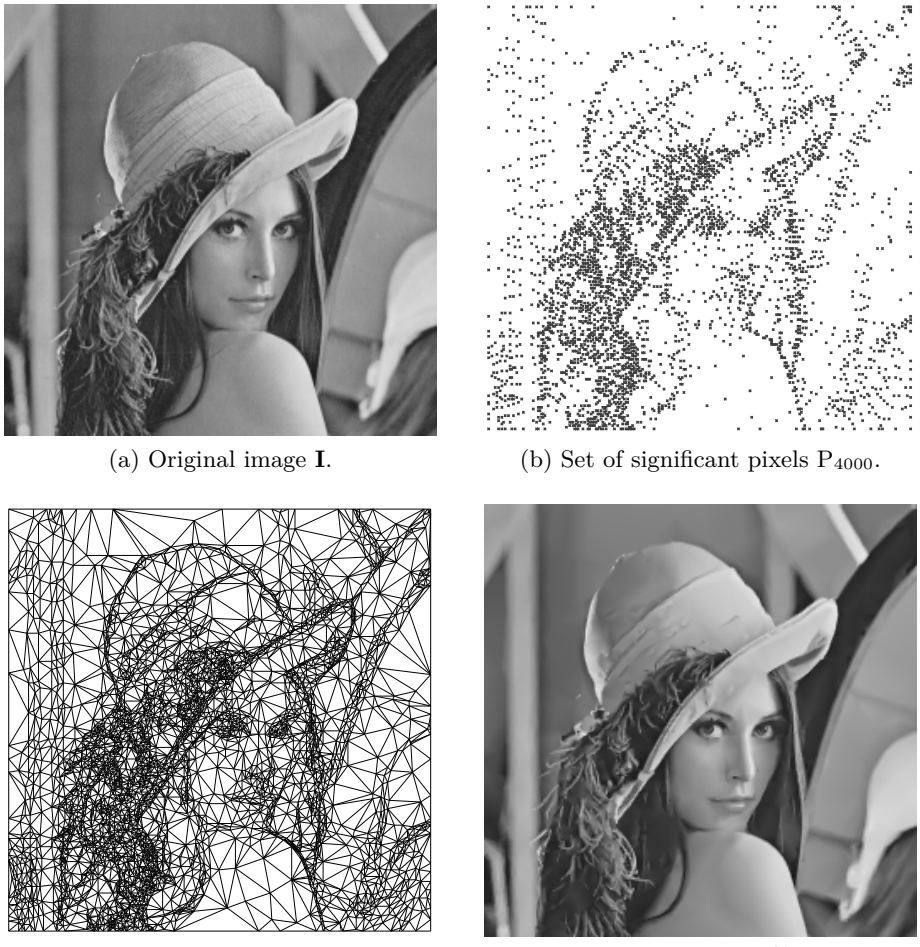
Output: approximation $\widetilde{\mathbf{I}} \approx \mathbf{I}$

1 Perturb P_n^* by the same rules applied at the encoder and compute the unique Delaunay triangulation $\mathcal{D}(\mathrm{P}_n^*)$.

2 Reconstruct $L_{\mathbf{I},\mathcal{D}}^* \in \mathcal{S}_{\mathcal{D}}$ from $\widetilde{\mathbf{I}}_{\mathrm{q}}^*$.

3 Reconstruct $\widetilde{\mathbf{I}}$.

An example of a compression of the well-known test image Lena is shown in Figure 3.4. The image resolution is 256×256, for a total of $|\mathrm{P}| = 65536$ pixels with a bit length $r = 8$, see 3.4(a). Consequently, the greyscale values of the luminances are in $\{0, \ldots, 255\}$, so that $\mathbf{I} \in \{0, \ldots, 255\}^{256 \times 256}$. We let the AT algorithm remove 61536 pixels, so that we obtain $4000 = n = |\mathrm{P}_n|$ significant pixels. Finally, we let $s = 5000$, so that a maximum of 5000 post-processing pixel exchanges take place. After acquiring the optimal luminances, the quantization is also performed with a step size of 8. The sparse set P_{4000} of significant pixels can be seen in 3.4(b) and its unique Delaunay triangulation in 3.4(c). Reconstructing $\mathbf{I}$ by linear splines over the Delaunay triangulation $\mathcal{D}(\mathrm{P}_{4000})$ yields the approximation $\widetilde{\mathbf{I}}$ seen in 3.4(d). $\widetilde{\mathbf{I}}$ admits a PSNR value of 34.3408 dB when compared to the original image $\mathbf{I}$.



(a) Original image $\mathbf{I}$.

(b) Set of significant pixels P_{4000}.

(c) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{4000})$.

(d) Reconstruction $\widetilde{\mathbf{I}}$ of $\mathbf{I}$.

Figure 3.4: AT reconstruction of Lena over a set of 4000 significant pixels.

Chapter 4

Graph Signal Processing

In the last chapter we have shown how to efficiently approximate digital images via linear splines over Delaunay triangulations. The approximation with linear splines is necessary for a unique reconstruction over a given triangulation, but also detrimental to the reconstruction of fine scale detail. To add these fines scale details back to the reconstruction, signal processing techniques for irregular data domains are essential.

Typically, digital signal processing deals with signals defined in continuous domains, which are sampled to get a digital representation for further processing. These signals are usually based on a domain such as space or time and represent evolution of a variable on a regular lattice (luminance on a two dimensional grid in the case of images). As such, a lot of research in conventional signal processing is directed at signals sampled over such rigid lattices, such as the DCT or DWT presented in Chapter 2. However, there are several applications that require a more flexible geometry. Prime examples for this are sensor, biological or neural networks, but also irregular, triangular image domains.

A popular method to add flexibility is to model the required domains by *graphs*. Graphs are simple structures that consist of some objects (possibly with associated data) and connections between them. As such, they are able to model pairwise relations between objects, for example the correlation of neighbouring pixels in an image. In recent years *spectral graph theory*, see [10], has been utilized to extend important mathematical tools and ideas from conventional Fourier analysis to a graph setting. While most of the early research focused on analyzing the underlying graphs, signals on graphs are of increasing interest. To this end, frequency spectra and expansion bases for a *graph Fourier transform* (GFT) [41] have been defined and applied to various problems as mentioned above. *Graph signal processing* (GSP) is the field that collects all these results, see [77], [80].

In this chapter we present the basic principles of graph signal processing, namely graphs and signals defined on them. Additionally, we extend concepts such as signal smoothness from conventional signal processing to signals defined on graphs and define a GFT.

4.1 Weighted Graphs

Graphs present a flexible way to model objects and pairwise relations between them. They comprise a set of *vertices* along with a set of *edges*. Vertices model objects, for example pixels in an image. Two vertices may be connected by an edge to convey some sense of relation.

Definition 4.1 *A finite* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *is an ordered pair of disjoint, finite sets, where* $\mathcal{V}$ *is a set of* vertices (nodes) *and* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *a set of* edges (links).

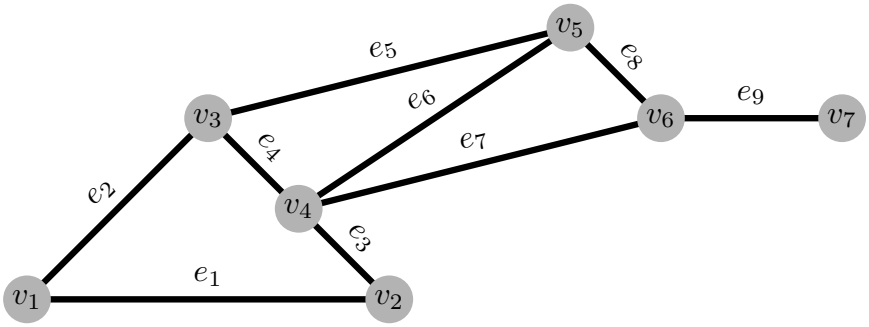
We denote a vertex by $v \in \mathcal{V}$ and the number of vertices by $N \in \mathbb{N}$, so that $\mathcal{V} = \{v_1, \dots, v_N\}$ with $|\mathcal{V}| = N$. The edges of a graph are denoted by $e \in \mathcal{E}$ and the number of edges by $M \in \mathbb{N}$, so that $\mathcal{E} \subseteq \{e = (v_i, v_j) : v_i, v_j \in \mathcal{V}\}$ and $|\mathcal{E}| = M$. We say that $\mathcal{E}$ defines a *topology* on $\mathcal{V}$. Depending on context, we alternatively denote a vertex by its index, i.e. $v_i \equiv i$, which translates to edges, i.e. $e = (v_i, v_j) \equiv (i, j)$. Two disctinct vertices $v_i, v_j \in \mathcal{V}$ connected by an edge $e \in \mathcal{E}$ are said to be *adjacent*, denoted by $v_i \sim v_j$, and e is said to be *incident* to v_i and v_j. Two distinct edges are incident if they share a common vertex. An example of a graph $\mathcal{G}_E$ with $\mathcal{V}_E = \{v_1, \dots, v_7\}$ and $\mathcal{E}_E = \{e_1, \dots, e_9\}$ is shown in Figure 4.1(a).

In order to characterize the pairwise, symmetric relationship between two distinct pixels, we introduce *weighted undirected graphs*.

Definition 4.2 *A* weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ *is an ordered triple, where* $\mathcal{V}$ *and* $\mathcal{E}$ *are defined as in Definition 4.1 and the edges are weighted via a symmetric, non-negative* weighted adjacency matrix $\mathbf{W} = [W_{i,j}]_{1 \leq i,j \leq N} \in \mathbb{R}^{N \times N}$. *In particular, an edge* $e = (v_i, v_j) \in \mathcal{E}$ *connecting vertices* $v_i, v_j \in \mathcal{V}$ *has the weight* $W_{i,j} = W_{j,i} > 0$. *If* v_i *and* v_j *are not adjacent we have* $W_{i,j} = W_{j,i} = 0$.

We do not allow loops, i.e. edges $e = (v_i, v_i)$, thus $W_{i,i} = 0$ for all $1 \leq i \leq N$. In this thesis, the weighted adjacency matrix $\mathbf{W}$ indicates similarities or dissimilarities between adjacent pixels in digital images. Their pairwise relationship is symmetric, hence the choice of undirected graphs. The edge set is symmetric, i.e. $v_i \sim v_j \Leftrightarrow v_j \sim v_i$ and $W_{i,j} = W_{j,i}$.

From here on out all graphs will be weighted and undirected unless otherwise stated. Additionally, we refer to the weighted adjecency matrix simply as *weight matrix*. An example of a weight matrix can be seen in Figure 4.1(b), where every edge in the previous example graph $\mathcal{G}_E$ has been weighted with weight 1.



(a) Example graph $\mathcal{G}_E$ consisting of seven vertices connected by 9 undirected edges.

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(b) Weighted adjacency matrix $\mathbf{W}_E$ of $\mathcal{G}_E$.

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(c) Degree matrix $\mathbf{D}_E$ of $\mathcal{G}_E$.

$$\begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

(d) Graph Laplacian $\mathbf{L}_E$ of $\mathcal{G}_E$.

Figure 4.1: An example graph $\mathcal{G}_E$ along with its weight, degree and graph Laplacian matrix.

The degree $d(v_i)$ of a vertex $v_i \in \mathcal{V}$ is the sum of weights of all edges incident to v_i, i.e.

$$d(v_i) \equiv d_i = \sum_{v_j \in \mathcal{N}_i} W_{i,j},$$

where $\mathcal{N}_i$ denotes the neighbourhood of v_i, that is all vertices adjacent to v_i. Note that the degree of a vertex may be computed by taking the sum of all elements in the i^{th} row of $\mathbf{W}$. All degrees constitute the *degree matrix* $\mathbf{D}$, which is the diagonal matrix that comprises the degrees of every vertex, i.e. $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$.

The main tool for graph signal processing as presented in [80] is the *graph Laplacian matrix*.

Definition 4.3 *Given a graph $\mathcal{G}$, its* (non-normalized) graph Laplacian matrix $\mathbf{L}$ *is defined as the difference between its degree matrix* $\mathbf{D}$ *and weight matrix* $\mathbf{W}$*, i.e.*

$$\mathbf{L} := \mathbf{D} - \mathbf{W}. \tag{4.1}$$

The degree matrix and graph Laplacian matrix of $\mathcal{G}_E$ are displayed in Figure 4.1(c) and 4.1(d) respectively. An extensive overview over the properties of the graph Laplacian can be found in [64] and [65]. In the following, we present the relevant facts for graph spectral processing on images.

The main building blocks of a GFT are the eigenvalues and eigenvectors of $\mathbf{L}$.

Proposition 4.1 *Given a graph $\mathcal{G}$, its graph Laplacian* $\mathbf{L}$ *is a real, symmetric and positive semi-definite matrix. As such, it admits a complete set of orthonormal eigenvectors* $\{\mathbf{u}_k\}_{k=0,1,\ldots N-1}$ *and associated real, non-negative eigenvalues* $\{\lambda_k\}_{k=0,1,\ldots N-1}$*, satisfying*

$$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k,$$

for all $k = 0, 1, ..., N-1$. □

Note that there is not necessarily a unique set of eigenvectors. We will assume throughout this thesis that a set of eigenvectors is chosen and fixed. Proposition 4.1 can alternatively be rewritten with matrices. Let $\mathbf{U} = [\mathbf{u}_0 | \ldots | \mathbf{u}_{N-1}] \in \mathbb{R}^{N \times N}$ be the matrix that comprises the eigenvectors of $\mathbf{L}$ as columns and $\mathbf{\Lambda} = \text{diag}(\lambda_0, \ldots, \lambda_{N-1}) \in \mathbb{R}^{N \times N}$ the matrix that contains the eigenvalues of $\mathbf{L}$ on its diagonal. Then we have

$$\mathbf{L}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}.$$

Note that all elements in a row of $\mathbf{L}$ sum to zero, thus 0 is a guaranteed eigenvalue. Moreover, the amount of such eigenvalues is linked to the number of connected components. A connected component C of $\mathcal{G}$ is a subset $C \subseteq \mathcal{V}$ with no edges between C and $\mathcal{V} \setminus C$, in which any two distinct vertices can be joined by a path that lies completely in C.

Proposition 4.2 *Given a graph $\mathcal{G}$, the multiplicity of eigenvalue* 0 *of* $\mathbf{L}$ *is equal to the number of connected components in $\mathcal{G}$. If $\mathcal{G}$ is connected, the eigenvalue $\lambda_0 = 0$ will appear exactly once, with the corresponding eigenvector being a constant vector, i.e.* $\mathbf{u}_0 = \frac{1}{\sqrt{N}}(1, \ldots, 1)^T$. □

We denote the full spectrum of $\mathbf{L}$ by $\sigma(\mathbf{L}) := \{\lambda_k\}_{k=0,1,\dots N-1}$. In this thesis we only regard fully connected graphs and sort the eigenvalues in ascending order, so that

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1}. \tag{4.2}$$

In some cases it may be beneficial to not have a constant vector as an eigenvector. Hence, another option is to normalize each weight $W_{i,j}$ by a factor of $\frac{1}{\sqrt{d_i d_j}}$, which yields the *normalized Graph Laplacian*.

Definition 4.4 *Given a graph $\mathcal{G}$, its* normalized graph Laplacian $\widetilde{\mathbf{L}}$ *is defined as*

$$\widetilde{\mathbf{L}} := \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}.$$

Even though the first eigenvalue is still zero, its associated eigenvector is not constant. We denote the eigenvectors of $\widetilde{\mathbf{L}}$ by $\{\widetilde{u}_l\}_{l=0,\dots,N-1}$ and its spectrum by $\sigma(\widetilde{\mathbf{L}}) := \{\widetilde{\lambda}_l\}_{l=0,\dots,N-1}$, where $\sigma(\widetilde{\mathbf{L}})$ satisfies

$$0 = \widetilde{\lambda}_0 < \widetilde{\lambda}_1 \leq \widetilde{\lambda}_2 \leq \cdots \leq \widetilde{\lambda}_{N-1} \leq 2.$$

Both the non-normalized and the normalized graph Laplacian are special cases of *generalized graph Laplacians*, see [4]. A generalized graph Laplacian of a graph $\mathcal{G}$ is a symmetric matrix $\mathbf{M}$ such that $M_{i,j} < 0$ whenever (v_i, v_j) is an edge of $\mathcal{G}$ and $M_{i,j} = 0$ if v_i and v_j are distinct and not adjacent.

4.2 Graph Signals

To extend the concept of signals to the graph setting, *graph signals* are defined as real valued functions that assign each vertex a value.

Definition 4.5 *A* graph signal $f : \mathcal{V} \to \mathbb{R}$ *defined on the vertices of a graph $\mathcal{G}$ maps every vertex $v \in \mathcal{V}$ to a value $f(v) \in \mathbb{R}$.*

A graph signal is represented by a vector $\mathbf{f} \in \mathbb{R}^N$, with the i^{th} value in $\mathbf{f}$ corresponding to the value of the i^{th} vertex v_i in $\mathcal{V}$. In this thesis we thus use the following terms equivalently $[\mathbf{f}]_i \equiv \mathbf{f}(i) \equiv \mathbf{f}(v_i)$. An example of a graph signal on the example graph $\mathcal{G}_E$ is provided in Figure 4.2. For graph signals $\mathbf{f} \in \mathbb{R}^N$ the graph Laplacian defines a difference operator, as it satisfies

$$(\mathbf{L}\mathbf{f})(i) = \sum_{v_j \in \mathcal{N}_i} W_{i,j}(\mathbf{f}(i) - \mathbf{f}(j)).$$

The Laplacian matrix can alternatively be constructed via the *incidence matrix*, see [4].

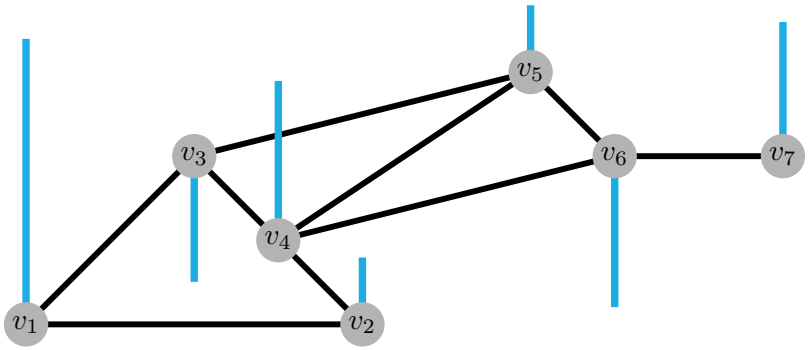


Figure 4.2: Graph signal on $\mathcal{G}_E$. The values at each vertex are represented by the blue bars, where the length depicts the magnitude and the direction the sign.

Definition 4.6 *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $\mathcal{E} = \{e_1, \dots, e_M\}$ and an arbitrary but fixed orientation for every edge $(i,j) \in \mathcal{E}$, the* incidence matrix $\mathbf{J} \in \mathbb{R}^{M \times N}$ *is defined by*

$$[\mathbf{J}]_{k,i} = \begin{cases} -1 & \text{if } e_k = (i,j), \\ 1 & \text{if } e_k = (j,i), \\ 0 & \text{otherwise,} \end{cases}$$

for $1 \leq k \leq M$ and $1 \leq i \leq N$.

As the incidence matrix depends on the order of edges $\mathcal{E} = \{e_1, \dots, e_M\}$, we assume an arbitrary but fixed order of edges is given for every weighted graph from here on out. The incidence matrix $\mathbf{J}_E$ of $\mathcal{G}_E$ is displayed in Figure 4.3(a). In order to construct $\mathbf{L}$ from $\mathbf{J}$, we let $\widehat{\mathbf{W}} \in \mathbb{R}^{M \times M}$ be the diagonal matrix with $\widehat{\mathbf{W}}_{k,k} = W_{i,j}$ for every edge $e_k = (i,j) \in \mathcal{E}$.

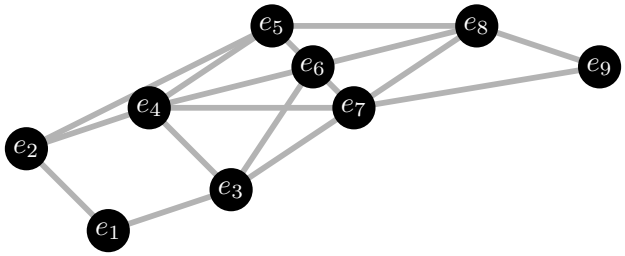
Proposition 4.3 *Let $\mathcal{G}$ be a graph with incidence matrix $\mathbf{J}$ and $\widehat{\mathbf{W}}$ as above. The graph Laplacian $\mathbf{L}$ of $\mathcal{G}$ is then given by*

$$\mathbf{L} = \mathbf{J}^T \widehat{\mathbf{W}} \mathbf{J}. \tag{4.3}$$

□

$$\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

(a) Incidence matrix $\mathbf{J}_E$ of $\mathcal{G}_E$.



(b) Line graph $\mathcal{G}_{E,\ell}$ of $\mathcal{G}_E$.

Figure 4.3: Incidence matrix and line graph of $\mathcal{G}_E$.

Note that the construction of $\mathbf{L}$ in (4.3) is indpendent of the edge orientation chosen and identical to the definition in (4.1). Given the incidence matrix of a graph, we are thus able to reconstruct $\mathbf{L}$ just from the diagonal entries of $\widehat{\mathbf{W}}$, which we alternatively arrange as an *edge weight vector*.

Definition 4.7 *Given a weighted Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with edge set $\mathcal{E} = \{e_1, \dots, e_M\}$, we define the* edge weight vector $\mathbf{w} \in \mathbb{R}^M$ *by*

$$[\mathbf{w}]_k := W_{i,j},$$

where $W_{i,j}$ is the weight of edge $e_k = (i, j) \in \mathcal{E}$.

The diagonal matrix $\widehat{\mathbf{W}}$ is reconstructed from the edge weight vector by $\widehat{\mathbf{W}} = \operatorname{diag}(\mathbf{w})$. In later chapters it will be useful to consider $\mathbf{w}$ as a graph signal on the *line graph* of $\mathcal{G}$.

Definition 4.8 *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, its* line graph $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$ *is an unweighted graph with vertex set $\mathcal{V}_\ell = \mathcal{E}$. Each node $v_e \in \mathcal{V}_\ell$ represents an edge $e \in \mathcal{E}$ and two nodes are adjacent iff their corresponding edges are incident in $\mathcal{G}$.*

The line graph $\mathcal{G}_{E,\ell}$ of $\mathcal{G}_E$ is displayed in Figure 4.3(b). For easier distinction we use the terms vertex and edge for the primal graph and node and link for the line graph. Note that the construction of $\mathcal{G}_\ell$ is independent of possible edge weights on $\mathcal{G}$ and only depends on its topology.

The edge weight graph signal $\mathbf{w} \equiv \mathbf{w}_\ell : \mathcal{E}_\ell \to \mathbb{R}$ simply assigns each node $e \in \mathcal{V}_\ell$ the weight w_e of its corresponding edge $e \in \mathcal{E}$.

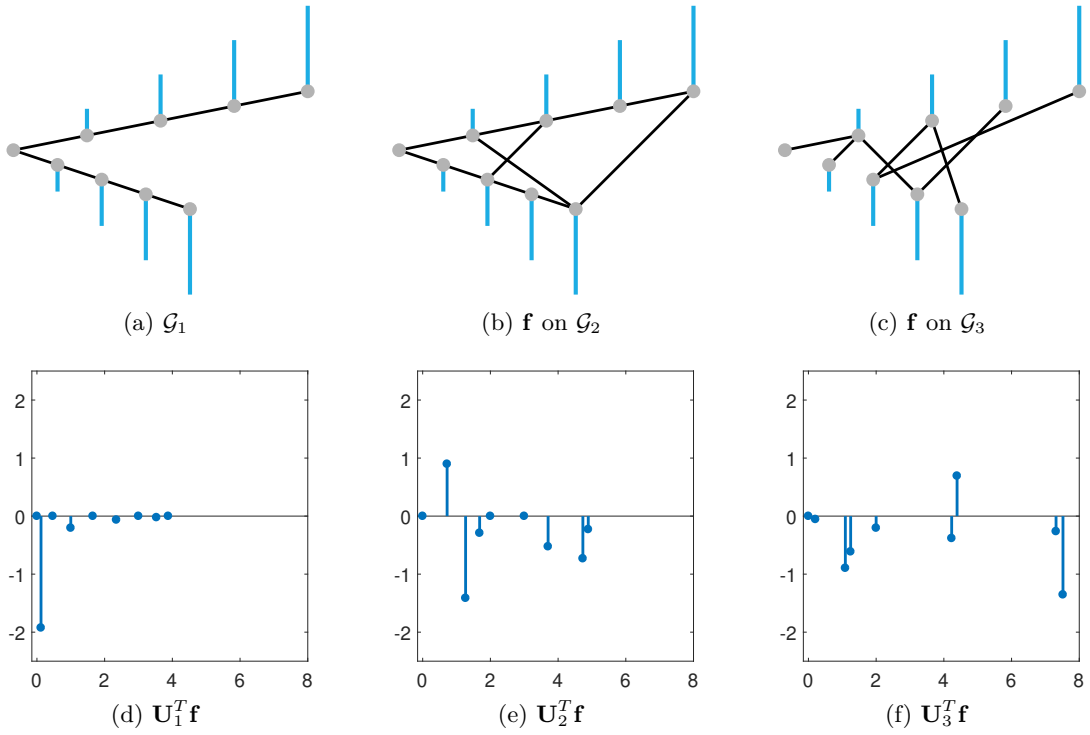
4.3 Graph Signal Smoothness

In conventional signal processing, smooth signals play an important role. They are *compressible*, as they can be closely approximated by a small number of coefficients in the spectral domain. More specifically, in classical Fourier analysis the eigenvalues of the complex exponentials $\left\{\mathrm{e}^{\mathrm{i}2\pi\xi t}\right\}_{\xi\in\mathbb{R}}$ convey a certain sense of frequency. They are the eigenfunctions of the Laplace-Operator and act as *elemental frequencies*. For values of ξ close to zero they are said to be *smooth*, or of low frequency. The larger $|\xi|$ gets, the coarser the signal becomes. In this sense, a smooth, low frequency signal is a function that is slowly osscilating, whereas coarse, high frequency functions oscillate much more rapidly.

When analyzing signals on graphs, their properties (especially the smoothness) depend on the intrinsic structure of the data domain. Here, it is represented by the weighted graph $\mathcal{G}$. A smooth graph signal $\mathbf{f}_s$ is a signal that varies slowly across the graph, whereas a coarse graph signal $\mathbf{f}_c$ oscillates more rapidly along the graph. To be more precise, we consider two vertices $v_i, v_j \in \mathcal{V}$ connected by an edge $e = (v_i, v_j) \in \mathcal{E}$. If the weight $W_{i,j} \gg 0$ is large, a smooth graph signal has similar values, i.e. $|\mathbf{f}_s(i) - \mathbf{f}_s(j)| \approx 0$. A coarse graph signal on the other hand could have dissimilar values.

An example of the influence underlying graphs can have on the smoothness of a graph signal is shown in Figure 4.4. It shows three graphs, $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, that share the same vertex set

$\mathcal{V} = \{v_1, \dots, v_9\}$. We are interested in the smoothness of a given graph signal given on each of them, namely $\mathbf{f} = (-1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1)^T$. What differs is the selection of edges, each with weight 1. While $\mathbf{f}$ is a smooth signal with respect to $\mathcal{G}_1$, as only vertices with similar values are adjacent, $\mathcal{G}_2$ increases the coarseness by adding edges between vertices with dissimilar values. $\mathcal{G}_3$ is the coarsest graph, as almost all edges connect vertices with dissimilar values.



(a) $\mathcal{G}_1$ (b) $\mathbf{f}$ on $\mathcal{G}_2$ (c) $\mathbf{f}$ on $\mathcal{G}_3$

(d) $\mathbf{U}_1^T\mathbf{f}$ (e) $\mathbf{U}_2^T\mathbf{f}$ (f) $\mathbf{U}_3^T\mathbf{f}$

Figure 4.4: An example of the importance of the underlying graph. $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ share the same vertex set and the same graph signal $\mathbf{f} = (-1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1)^T$ is given on them. The increasing coarseness can be detected visually, via the graph Laplacian form, as $\mathbf{f}^T\mathbf{L}_1\mathbf{f} = 0.5 < \mathbf{f}^T\mathbf{L}_2\mathbf{f} \approx 7.1 < \mathbf{f}^T\mathbf{L}_3\mathbf{f} \approx 18.4$, or through the graph spectral representations. After applying the GFT the information on $\mathcal{G}_1$ is compacted in the low frequencies, while the energy is distributed more evenly for coarser graphs.

In order to analyze the smoothness of a signal with respect to a graph, objective measures have been developed. Differential geometry provides tools to analyze continuous signals on given geometric structures. Therefore, we turn to *discrete calculus* on graphs in order to extend these intuitions and ideas to the graph setting, see [30], [95].

The smallest building block is the directional *edge derivative*. It measures the variation of a signal along an edge at a vertex.

Definition 4.9 *Given a graph $\mathcal{G}$ along with a graph signal $\mathbf{f}$, the* edge derivative *of an edge $e = (i, j) \in \mathcal{E}$ at vertex $v_i \in \mathcal{V}$ is defined as*

$$\left.\frac{\partial \mathbf{f}}{\partial e}\right|_{v_i} := \sqrt{W_{i,j}}(f(j) - f(i)).$$

Accumulating all edge derivatives at a vertex in a vector results in the *graph gradient*.

Definition 4.10 *Given a graph $\mathcal{G}$ along with a graph signal $\mathbf{f}$, the* graph gradient *of $\mathbf{f}$ at a vertex $v_i \in \mathcal{V}$ is defined as*

$$\nabla_i \mathbf{f} := \left[\left. \frac{\partial \mathbf{f}}{\partial e} \right|_{v_i} \right]_{e \in \mathcal{E} \text{ so that } e=(v_i,\cdot)}.$$

The local variation *at vertex v_i is defined as its norm, i.e.*

$$\begin{aligned} \|\nabla_i \mathbf{f}\|_2 &:= \left(\sum_{e \in \mathcal{E},\ e=(v_i,\cdot)} \left(\left. \frac{\partial \mathbf{f}}{\partial e} \right|_{v_i} \right)^2 \right)^{\frac{1}{2}} \\ &= \left(\sum_{e \in \mathcal{E},\ e=(v_i,\cdot)} W_{i,j} \left(f(j) - f(i) \right)^2 \right)^{\frac{1}{2}}. \end{aligned} \tag{4.4}$$

The local variation gives an indication of the local smoothness of $\mathbf{f}$ around vertex v_i. $\|\nabla_i \mathbf{f}\|_2$ tends to be small if $\mathbf{f}$ has similar values at a vertex and its neighbourhood or if dissimilar values are connected via an edge with small weight. For notions of global smoothness we consider the *discrete p-Dirichlet form*.

Definition 4.11 *Given a graph $\mathcal{G}$ along with a graph signal $\mathbf{f}$, the* discrete p-Dirichlet form *of $\mathbf{f}$ is for $p \in \mathbb{N}$ defined as*

$$S_p(\mathbf{f}) := \frac{1}{p} \sum_{v_i \in \mathcal{V}} \|\nabla_i \mathbf{f}\|_2^p = \frac{1}{p} \sum_{v_i \in \mathcal{V}} \left(\sum_{v_j \in \mathcal{N}_i} W_{i,j} \left(f(j) - f(i) \right)^2 \right)^{\frac{p}{2}}. \tag{4.5}$$

If $p = 1$, then $S_1(\mathbf{f})$ is the *total variation* of $\mathbf{f}$ with respect to the underlying graph. For later chapters it will be useful to measure the global smoothness by setting $p = 2$.

Definition 4.12 *Given a graph $\mathcal{G}$ along with a graph signal $\mathbf{f}$, the* graph Laplacian quadratic form *is defined as*

$$\begin{aligned} S_2(\mathbf{f}) &= \frac{1}{2} \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}_i} W_{i,j} (f(j) - f(i))^2 \\ &= \sum_{(i,j) \in \mathcal{E}} W_{i,j} (f(j) - f(i))^2 = \mathbf{f}^T \mathbf{L} \mathbf{f}. \end{aligned} \tag{4.6}$$

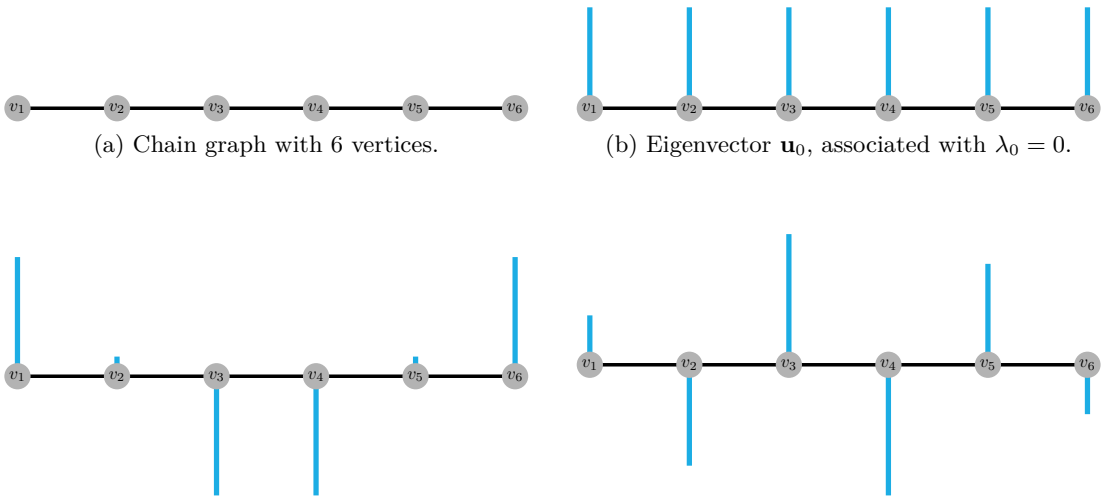
Note from (4.6) that $\mathbf{f}^T \mathbf{L} \mathbf{f} = 0$ iff $\mathbf{f}$ is constant along all vertices, i.e. $\mathbf{f}(i) = c \in \mathbb{R}$ for all $1 \leq i \leq N$. More generally, we observe that $\mathbf{f}^T \mathbf{L} \mathbf{f}$ weighs the differences of adjacent values of a graph signal along an edge by its corresponding edge weight. In smooth signals, an edge with large weight will connect two vertices with similar values, whereas vertices with dissimilar values are connected by an edge with low weight. This results in small values of $\mathbf{f}^T \mathbf{L} \mathbf{f}$ if the graph signal is smooth with respect to the graph and large values if it is coarse. The graph Laplacian quadratic form defines a semi-norm $\|\cdot\|_{\mathbf{L}}$ via

$$\|\mathbf{f}\|_{\mathbf{L}} = \left\| \mathbf{L}^{\frac{1}{2}} \mathbf{f} \right\|_2 = \sqrt{\mathbf{f}^T \mathbf{L} \mathbf{f}} = \sqrt{S_2(\mathbf{f})}.$$

In addition to this, $S_2(\mathbf{f}) = \mathbf{f}^T\mathbf{L}\mathbf{f}$ is cheaply computed. As such, it is the preferred indicator for the smoothness of a signal with respect to the underlying graph. The above considerations are confirmed by Figure 4.4, where the values of the graph Laplacian quadratic form of $\mathbf{f}$ in regard to each graph are given. Note that the value increases as the graph gets coarser, from $\mathbf{f}^T\mathbf{L}_1\mathbf{f} = 0.5$ to $\mathbf{f}^T\mathbf{L}_3\mathbf{f} \approx 18.4$.

4.4 The Graph Fourier Transform

Similarly to classical Fourier analysis, the eigenvectors of the graph Laplacian carry a sense of frequency. Graph Laplacian eigenvectors associated with small eigenvalues correspond to low frequencies, i.e. they are smoother and vary only slowly across the underlying graph. As the eigenvalues grow, the associated Laplacian eigenvectors become coarser. An example of this can be seen in Figure 4.5, where we show some eigenvectors of the graph Laplacian of the simple chain graph with constant edge weight 1 in 4.5(a). While the first eigenvector $\mathbf{u}_0$ associated with λ_0 is constant, the variations along edges increase for the third eigenvector $\mathbf{u}_2$. The eigenvector $\mathbf{u}_5$ corresponding to the largest eigenvalue λ_5 is very coarse, with large variations along every edge.



(a) Chain graph with 6 vertices.

(b) Eigenvector $\mathbf{u}_0$, associated with $\lambda_0 = 0$.

(c) Eigenvector $\mathbf{u}_2$, associated with $\lambda_2 = 1$.

(d) Eigenvector $\mathbf{u}_5$, associated with $\lambda_5 \approx 3.73$.

Figure 4.5: Increasing frequency of the eigenvectors of the graph Laplacian demonstrated on a chain graph.

This subjective impression is validated through the Courant-Fischer Theorem, see [4]. By applying the theorem to the graph Laplacian $\mathbf{L}$, the eigenvalues and eigenvectors can be defined iteratively via the Rayleigh quotient.

Proposition 4.4 *Given a graph $\mathcal{G}$, its spectrum $\sigma(\mathbf{L}) := \{\lambda_0, \ldots, \lambda_{N-1}\}$ is given by*

$$\lambda_0 = \min_{\substack{\mathbf{f}\in\mathbb{R}^N \\ \|\mathbf{f}\|_2=1}} \mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{4.7}$$

$$\lambda_k = \min_{\substack{\mathbf{f}\in\mathbb{R}^N \\ \|\mathbf{f}\|_2=1 \\ \mathbf{f}\perp\mathrm{span}(\mathbf{u}_0,\ldots,\mathbf{u}_{k-1})}} \mathbf{f}^T\mathbf{L}\mathbf{f} \tag{4.8}$$

for $1 \leq k \leq N-1$. The eigenvector $\mathbf{u}_k$ is the minimizer of the k^{th} problem. □

Proposition 4.4 links the eigenvalues and eigenvectors to the graph Laplacian quadratic form, which measures the smoothness of a graph signal. We begin with the eigenvector corresponding to $\mathbf{u}_0 = 0$, which is a constant vector with norm 1 in (4.7). As the index k increases, the eigenvector $\mathbf{u}_k$ minimzing (4.8) has a larger value for $\mathbf{f}^T\mathbf{L}\mathbf{f}$ and is thus coarser than the previous eigenvectors corresponding to lower indices.

The classical Fourier transform decomposes a signal $f \in \mathrm{L}^1(\mathbb{R})$ into its elemental frequencies. Likewise, the *Graph Fourier Transform* (GFT) decomposes a graph signal $\mathbf{f} \in \mathbb{R}^N$, given on the vertices of a graph $\mathcal{G}$, into its elemental frequencies. In this context the *elemental frequencies* are given by the eigenvectors $\mathbf{u}_k$ with *frequency* λ_k for $0 \leq k \leq N-1$. Depending on context, we designate both λ_k and k as the frequency of an elemental frequency. To this end, we analogously define the GFT as the expansion of a graph signal in terms of the eigenvectors of the corresponding graph Laplacian.

Definition 4.13 *Given a graph $\mathcal{G}$ along with a graph signal $\mathbf{f}$, the* graph Fourier transform $\hat{\mathbf{f}} \in \mathbb{R}^N$ *is defined as*

$$\hat{\mathbf{f}} := \mathbf{U}^T\mathbf{f}. \tag{4.9}$$

In contrast to the classical setting, where the Fourier transform $\hat{\mathbf{f}} \in \mathscr{C}(\mathbb{R})$ is defined on the whole real line, the GFT is only defined on discrete values, namely the spectrum $\sigma(\mathbf{L})$ of $\mathbf{L}$. Therefore, in later analysis of signals, we only consider a discrete set of frequencies. The Fourier coefficient of a specific eigenvalue $\lambda_k \in \sigma(\mathbf{L})$ for $0 \leq k \leq N-1$ is given by

$$\hat{\mathbf{f}}(\lambda_k) \equiv \hat{\mathbf{f}}(k) = \langle \mathbf{f}, \mathbf{u}_k \rangle = \sum_{i=1}^{N} \mathbf{f}(i)\mathbf{u}_k(i)$$

The GFT as defined in (4.9) generalizes the two-dimensional DCT in the following sense, as shown in [94]. Consider a graph $\mathcal{G}$ in which the vertices are spatially arranged in a regular two-dimensional grid. Each vertex is connected to its direct horizontal and vertical neighbours with an edge that has weight 1. Then the GFT degenerates to the two-dimensional DCT. This is the reason we use the non-normalized graph Laplacian for now, as its GFT is at least as good as the DCT.

In [94] it is further shown that the GFT is optimal in decorrelating a certain class of signals. Consider a signal $\mathbf{f} \in \mathbb{R}^N$ following a zero mean Gaussian Markov Random Field with respect to

a graph $\mathcal{G}$ and precision matrix Q. Then the Laplacian matrix $\mathbf{L}$ of $\mathcal{G}$ corresponds to the precision matrix Q and the graph Fourier transform is the KLT.

The *inverse graph Fourier transform* simply reverses the graph Fourier transform.

Definition 4.14 *Given a graph $\mathcal{G}$ along with a Fourier coefficients $\hat{\mathbf{f}} \in \mathbb{R}^N$, the* inverse graph Fourier transform *is defined as*

$$\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}. \tag{4.10}$$

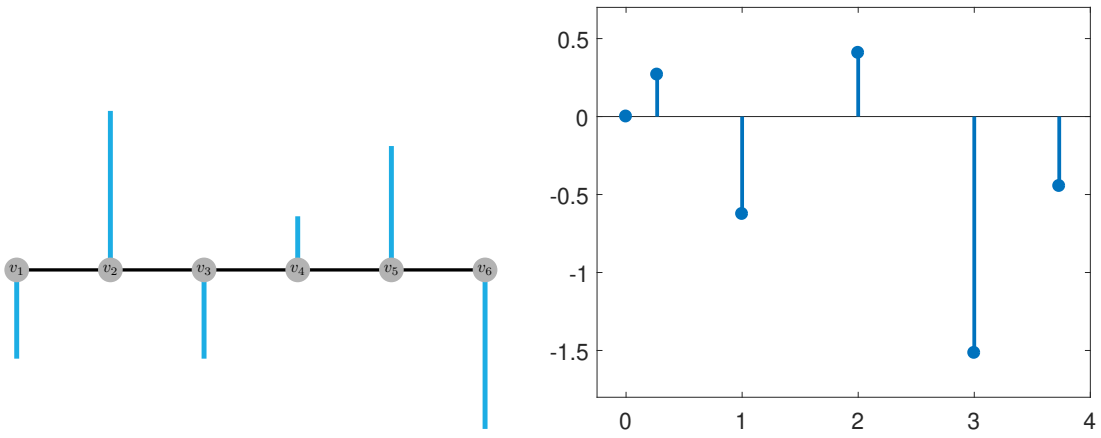
Recall that $\mathbf{U}$ is an orthonormal matrix, so that

$$\mathbf{f} := \mathbf{U}\mathbf{U}^T\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}.$$

Thus, a graph signal $\mathbf{f}$ may be reconstructed via its graph Fourier coefficients $\hat{\mathbf{f}}$. This can be understood as a weighted sum, where the graph Fourier coefficient $\hat{\mathbf{f}}(k)$ weighs the contribution of its associated elemental frequency $\mathbf{u}_k$. In particular, a specific graph signal value is reconstruced via

$$\mathbf{f}(i) = \sum_{k=0}^{N-1} \hat{\mathbf{f}}(\lambda_k)\mathbf{u}_k(i)$$

for $1 \leq i \leq N$.



(a) Graph signal $\mathbf{f}$ on a chain graph in the vertex domain.

(b) The transformed signal $\hat{\mathbf{f}}$ in the graph spectral domain.

Figure 4.6: Equivalent representation of a graph signal in the vertex and graph spectral domain. On the left is the standard graph signal that resides on the vertices of a graph. The same signal transformed to the graph spectral domain is shown on the right, where the Fourier coefficient $\hat{\mathbf{f}}(k)$ of each eigenvector $\mathbf{u}_k$ is mapped over the associated eigenvalue λ_k for $0 \leq k \leq 5$. Note that the graph signal in (a) has mean zero. Consequently, we have $\hat{\mathbf{f}}(\lambda_0) = 0$.

These two transforms give us two equivalent ways to represent a graph signal in different domains: (i) the vertex domain, which corresponds to the conventional time or spatial domain, and (ii) the graph spectral domain, which corresponds to the frequency domain in conventional signal processing. An example of the two representations in different domains is depicted in

Figure 4.6, where a graph signal on the chain graph in 4.5(a) is converted to the graph spectral domain.

With the Definition in (4.9) the Parseval relation holds, see [81], i.e. for any graph signals $\mathbf{f}, \mathbf{g} \in \mathbb{R}^N$ we have

$$\langle \mathbf{f}, \mathbf{g} \rangle = \langle \hat{\mathbf{f}}, \hat{\mathbf{g}} \rangle.$$

Consequently, the energy of a signal remains unchanged by the GFT, since

$$\|\mathbf{f}\|_2^2 = \langle \mathbf{f}, \mathbf{f} \rangle = \langle \hat{\mathbf{f}}, \hat{\mathbf{f}} \rangle = \|\hat{\mathbf{f}}\|_2^2. \tag{4.11}$$

Even though the total energy remains unchanged, its concentration may change depending on the underlying graph. An example of this is seen in Figure 4.4, where the graph Fourier transform of a signal is shown on three increasingly coarser graphs. While most of the energy on the smoothest graph is contained in one coefficient associated with a low frequency, the energy is more evenly distributed on the coarser graphs. Our focus from now on will be to construct graphs so that a given signal is compressible on them. Following the discussion above, it will be important to select weights that weigh edges between pixels with large intensity differences lowly, while allowing large weights between similar pixels.

Chapter 5

Combining Adaptive Thinning and Graph Signal Processing

We have introduced the method of adaptive thinning in Chapter 3, that efficiently reconstructs images with linear splines over a Delaunay triangulation of carefully selected significant pixels. Afterwards, Chapter 4 introduced graph signal processing as a tool to define frequency spectra on irregular image domains.

In this chapter we show how to combine both approaches and propose a post-processing scheme, named simply *ATGSP*, for the adaptive thinning algorithm. It utilizes graph spectral processing to improve the compression by adaptive thinning. To be more specific, we add additional information on triangles that are reconstructed poorly and have visual impactful information missing. Graphs are constructed on them and the individual triangle signals are treated as graph signals and compressed.

Due to the reconstruction with linear splines, most visually detrimental triangles will be *textured*. Textures provide an important contribution to the visual quality, especially the repetetive structure of texture patterns is crucial for a good visual quality, see [83]. Therefore, we adapt tools from the literature and develope techniques that construct weighted graphs to capture and enhance these texture patterns. To this end, the blocks are classified as either smooth, textured with a dominant principal gradient or complexly textured with the *structure tensor*. We show how to construct graphs so that the given signal is smooth in regard to them.

For now we focus on the theoretical aspect, before discussing the practical implementation in Chapter 6. An overview of the proposed ATGSP scheme can be seen in Figure 5.1. The individual steps will be explained in more detail in the following sections.

5.1 Related Work on Graph Spectral Image Compression

Before we introduce graph signal processing on triangular image blocks, we describe alternative techniques found in the literature for image compression utilizing GSP.

While it has already been used extensively in image processing such as *image restoration*, see for example [7], [44], [87], [92], *image filtering*, see for example [34], [66], [91], and *image segmentation*, see for example [53], [93], [59], we focus on *image compression* utilizing the GFT in this section. For a more extensive overview of all the topics mentioned above, we refer to the survey paper [8].

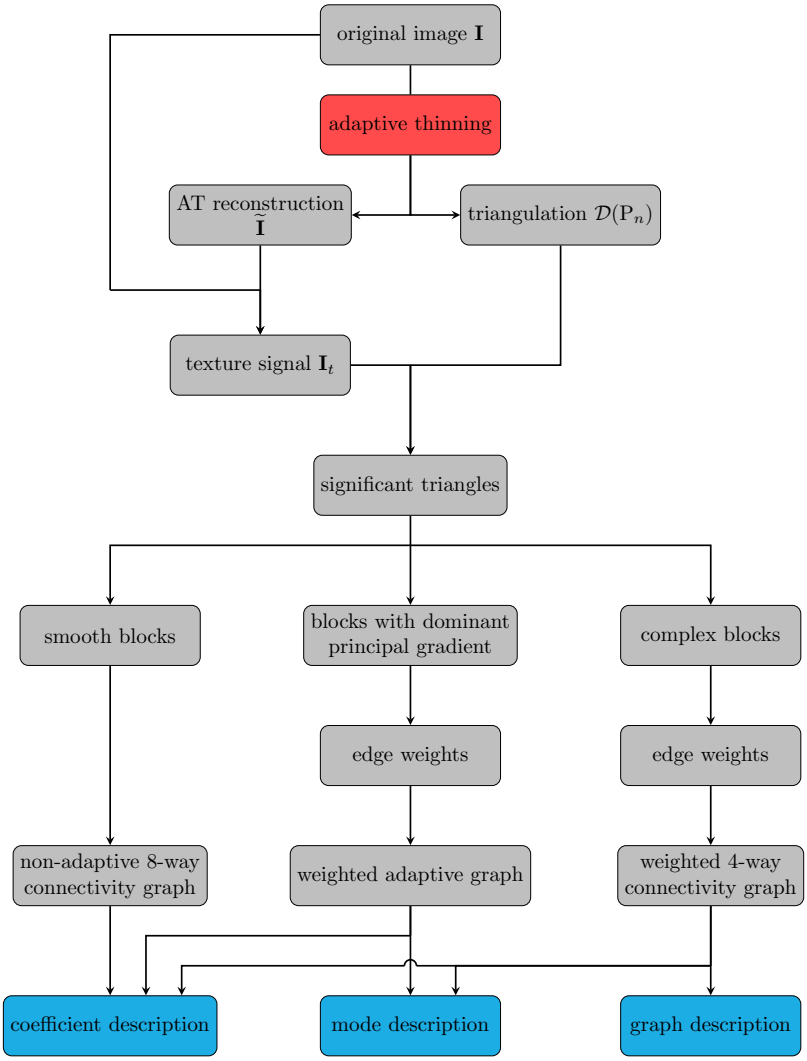


Figure 5.1: Overview of the proposed ATGSP compression scheme. A texture signal is derived from the AT reconstruction. Significant triangles are classified into three classes and their signals compressed. For textured image blocks adaptive graphs are constructed. Each significant signal is transformed via the GFT. Based on the classification, the coefficient, mode and graph description have to be encoded.

The basic approach of image compression via GSP is fairly straightforward. Here, we only give a brief overview and refer to Section 5.3 for a more detailed discussion. Each of the approaches introduced in this section splits an image into $\sqrt{N} \times \sqrt{N}$ pixel blocks, reducing the computational cost. This is the fundamental difference to our proposed method, where the image blocks are given as triangles of a triangulation generated by AT. Additionally, we aim to add extra detail only to selected areas of the AT reconstruction, namely those with notable visual distortions. As a result, we assume low bit rates and signals with a high textural content, see Sections 5.2 and 5.4.

Given an image signal $\mathbf{f} \in \mathbb{R}^N$ on a set of pixels $\mathrm{P}_\mathbf{f}$, an associated weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is constructed. The vertices $\mathcal{V}$ of the graph are the pixels $\mathrm{P}_\mathbf{f}$. A graph signal $\mathbf{f}_\mathcal{G}$ is constructed from the image signal by assigning each vertex the intensity of its corresponding pixel, i.e. $\mathbf{f}_\mathcal{G}(v) = \mathbf{f}(p)$ if vertex $v \in \mathcal{V}$ corresponds to pixel $p \in \mathrm{P}_\mathbf{f}$.

To encode the graph signal $\mathbf{f}$, it is transferred to the graph spectral domain via the GFT, where suitable filtering methods can be applied. At the decoder, the quantized coefficients $\hat{\mathbf{f}}_{\mathrm{ind}}$

are transferred back to the vertex domain via the inverse GFT, which results in an approximation $\tilde{\mathbf{f}} \approx \mathbf{f}$. As in conventional image processing, the key to good compression is to compact most energy into a few coefficients.

Since the GFT is fully defined by $\mathbf{L}$, which in turn depends on the choice of $\mathbf{W}$, it is of utmost importance to select a suitable edge set $\mathcal{E}$ and weights $\mathbf{W}$ yielding the best compression performance. In the graph setting, this means to select a graph $\mathcal{G}$ so that the image signal $\mathbf{f}$ is smooth with regard to it.

The approach in [79] designs an edge-adaptive transform, where the edge weights are in the set $\{0, 1\}$. Edge detection is applied to each block and generates a binary edge map. A graph is generated from the edge map by connecting each vertex to its four immediate neighbours only if there is no sharp edge between them. This implies a strong correlation between the connected pixels. An edge-adaptive transform is then generated by constructing $\mathbf{L}$ from the constructed weighted graph. The encoder compares the rate-distortion performance of the edge-adaptive transform with that of the DCT and selects the method that yields the lower rate-distortion cost. Selecting the weights in $\{0, 1\}$ leads to disconnected graphs, which increases the coefficient transmission cost. Additionally, the triangles given by the AT algorithm already give a good approximation of edges, yielding an extra edge description unnecessary.

In [45], the authors optimize the GFT for piecewise smooth images. They consider both weighted and unweighted transforms, where the weighted GFT imposes edge weights to be in the discrete set $\{w_\mathrm{o}, 1\}$. The optimal edge weight $w_\mathrm{o} \in (0, 1]$ for weakly correlated pixels is derived so that the resulting GFT approximates the KLT on a piecewise smooth model signal. For each block signal a *graph learning* problem is defined. It optimizes the transformation in rate-distortion terms. A further reduction of the transmission and complexity cost is achieved via a lookup table that stores the most popular graphs. At the encoder it compares the performance of a low-resolution GFT with a high resolution GFT and selects the more efficient transform. As this approach is optimized for piecewise smooth images, it is not compatible with our setting. We do, however, generalize the derivation of an optimal edge weight w_o for weakly correlated pixels to natural images in Section 5.8.

An optimization problem is also formulated in [33] to minimize the rate-distortion terms, but explicitly accounts for the graph-transmission cost. They treat the weights as a signal that lies on the line graph. Hence, they are not limited to discrete edge weights and are thus able to reflect the pairwise relationship between pixels better. While this method is designed for natural images, it assumes a high bit rate, which does not always transfer to our setting. Nevertheless, we adapt the optimization problem to our needs in section 5.7.1, but compare it with a new method assuming low bit rates.

The approach in [75] utilizes the *structure tensor* as a tool to summarize the distribution of the gradient. Blocks with a dominant principal gradient are clustered into groups depending on the angle of the principal gradient. For each cluster, a graph template is designed to reflect the structure of the image on its blocks. As we operate on natural images, not all image blocks have a

dominant principal gradient. We adapt this approach for triangular image blocks with a dominant gradient in Section 5.6.

A similar approach is chosen in [69], which introduces a new class of graph transforms named *graph template transform*. It exploits a priori information known about a block and represents it using a graph template. Building upon this information, it imposes a sparsity pattern on the graph Laplacian to minimize the graph transmission cost and approximates the KLT. But while it is effective at constructing a graph so that $\mathbf{f}$ is smooth in regard to it, encoding it is still inefficient.

5.2 Geometric and Textural Components of an Image

An image can be regarded as a composition of three parts:

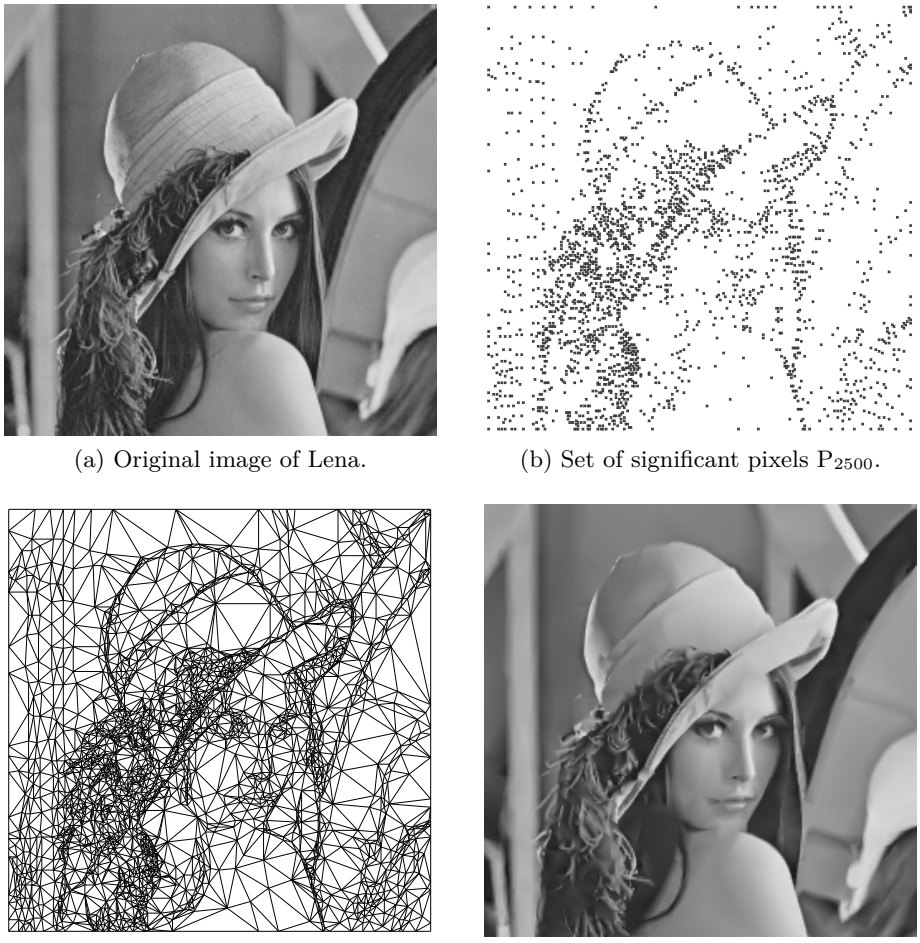
- the *geometric part* of an image comprises the large geometrical features of an image,
- the *textural part* is the part of an image that contains the fine scale details (*textures*),
- and *noise* is a deviation of a signal from its true value by a small, random quantity.

Noise is introduced by external factors, usually during image acquisition, and is present in many images. To mitigate the level of noise, algorithms decompose an image into the true signal part and a noise part. This may be done for example by spatial, transform or frequency filtering methods, dictionary learning methods or hybrid methods. For a comprehensive overview of denoising techniques we refer to [37]. From here on out we assume that a given image is essentially noise-free.

Thus, we are only interested in decomposing an image into its geometric and textural components. An exact, objective distinction between these two components is difficult, as the definition of each component is vague and depends on the image at hand. Depending on scale, the textures contained in one image may be the geometric component of another one at a smaller scale. Typically, the main geometrical features that comprise the geometric component are separated by decisive edges and generally correspond to low frequencies in the frequency domain. Textures are usually of a local, periodically oscillating nature and generally correspond to high frequencies.

For example, if we again consider the image Lena in Figure 5.2(a), the main geometrical features are Lena in the foreground, comprising separate minor objects like her hat, face and shoulder, as well as some separate pieces of furniture in the background. Conversely, fine scale details such as the structure of the straw hat, its feathers and her hair constitute the textural part.

Having made this intuitive disctinction between the geometry and textures of an image, we consider a reconstruction by AT. As discussed in Chapter 3, we approximate an image $\mathbf{I}$ by linear splines over the Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$ of an adaptively chosen pixel set P_n, for $4 \leq n \leq |\mathrm{P}|$. Since the most significant pixels in P_n are selected to minimize the loss of PSNR, pixel-pairs straddling sharp edges are usually contained in P_n. On the other hand, pixels in regions with only small variations in intensity values are more likely to be eliminated during the thinning process. Therefore, significant pixels are expected to be concentrated around the edges of an



(a) Original image of Lena.

(b) Set of significant pixels P_{2500}.

(c) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{2500})$.

(d) AT reconstruction $\widetilde{\mathbf{I}}$.

Figure 5.2: Original image of Lena along with a set of significant pixels, its Delaunay triangulation and the corresponding AT reconstruction.

image. As a consequence, a reconstruction by AT is able to capture the geometric component of an image very well.

The same is not true for fine scale details. While the choice of linear splines as an approximation space is essential due to the unique interpolation property on triangulations, it is detrimental to the reconstruction of textures. Since the significant pixels are concentrated around edges, there are potentially large regions with a low concentration of them, that are consequently covered by large triangles. For a sparse set P_n with $n \ll |\mathrm{P}|$, comprising only few significant pixels, these regions may very well contain visually significant textures. Linear splines are inherently unsuitable to represent textures, since approximating $\mathbf{I}$ with linear splines over large triangles will flatten out most fine scale details. Accordingly, a reconstruction by AT is not able to capture most textures of an image.

This is supported by the exemplary AT reconstruction over $n = 2500$ significant pixels shown in Figure 5.2. As predicted, they are concentrated around sharp edges in 5.2(b). Heavily textured regions such as the straw hat or her hair contain only few significant pixels. Therefore, the Delaunay triangulation $\mathcal{D}(\mathrm{P}_{2500})$ in figure 5.2(c) covers those regions with large triangles. When comparing the reconstruction in figure 5.2(d) to the original image, these regions look flat and

devoid of any structure. The feathers of the hat are an example of the vague definition of textures. Even though viewing the image one would classify them as small scale details, single feathers are distinct enough that the AT algorithm maintains a high concentration of pixels in this area.

As the AT algorithm approximates the geometric components of an image very well, our goal from here on out is to add textures for a better visual quality. Since the PSNR is a bad representation of visual quality, especially regarding textures, we evaluate the image quality via the SSIM as introduced in Section 2.1 for the rest of this thesis. To decompose the image $\mathbf{I}$ into its geometric and textural part, we assume the reconstruction $\widetilde{\mathbf{I}}$ by AT to constitute the geometric component. We consequently define the difference between $\mathbf{I}$ and $\widetilde{\mathbf{I}}$ as the textures.

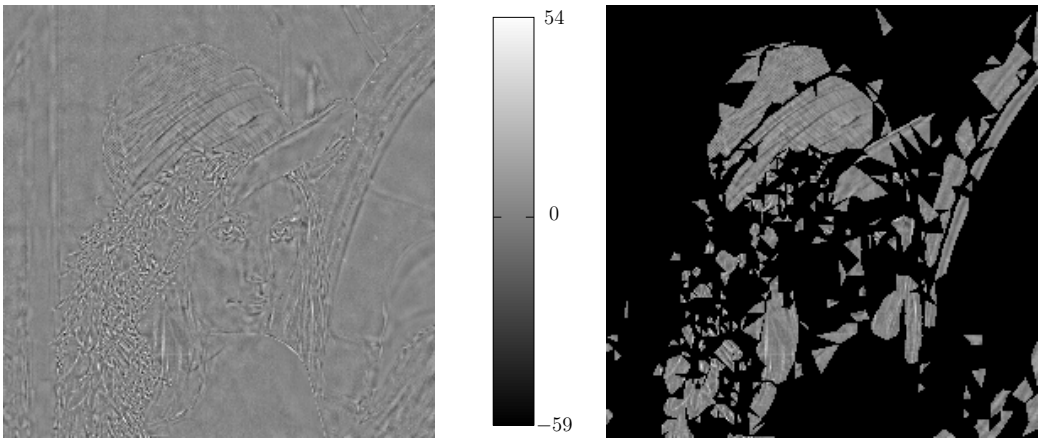
Definition 5.1 *Given an image* $\mathbf{I}$ *along with an reconstruction by adaptive thinning* $\widetilde{\mathbf{I}}$*, we denote the* textural component *of an image by* $\mathbf{I}_t$ *and define it as*

$$\mathbf{I}_t := \mathbf{I} - \widetilde{\mathbf{I}}. \tag{5.1}$$

We focus on the compression of visually significant sections of the textural component $\mathbf{I}_t$. Note that while $\mathbf{I}_t \in \{-255, \dots, 0, \dots, 255\}^{\mathrm{P}}$ has a potentially large range, due to the definition of $\mathbf{I}_t$ and the nature of the approximation $\widetilde{\mathbf{I}}$ its values will generally be contained in a limited range around 0. This is advantageous for quantization in Section 6.1.4, as a smaller quantization support reduces the quantization error. Figure 5.3(a) shows $\mathbf{I}_t$ as the difference between $\mathbf{I}$ and $\widetilde{\mathbf{I}}$ in figures 5.2(a) and 5.2(d) respectively. Even though the intensities are scaled up to the full intensity range, the background is mainly smooth, whereas the hat, hair and feathers are coarse. Additionally, the original structure of the straw hat is mostly contained in $\mathbf{I}_t$.

The AT reconstruction $\widetilde{\mathbf{I}}$ relies on the Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$. Therefore, we utilize the triangles in it as the basic image blocks for further processing. Employing techniques from GSP, we add extra information on triangles $\mathrm{T} \in \mathcal{D}(\mathrm{P}_n)$ to approximate $\mathbf{I}_t$ on them. Unfortunately, a triangulation of a point set with n vertices consists of $|(\mathcal{D}(\mathrm{P}_n))| \sim 2n$ triangles due to the Euler characteristic, see [13]. Adding information for every triangle would clearly be infeasible. Rather, we select a set of *most significant triangles* $\mathcal{T}_{\mathrm{sig}} \subset \mathcal{D}(\mathrm{P}_n)$.

We will show how to select a specific set $\mathcal{T}_{\mathrm{sig}}$ in Section 6.1.1 based on the desired reconstruction quality. Generally, triangles in $\mathcal{T}_{\mathrm{sig}}$ add the most visual quality among all triangles in $\mathcal{D}(\mathrm{P}_n)$. They are usually large and contain important textures, which increases the visual quality when they are combined with $\widetilde{\mathbf{I}}$. Figure 5.3(b) displays the most significant textured triangles from $\mathbf{I}_t$ in 5.3(a) for a medium quality. Most triangles in $\mathcal{T}_{\mathrm{sig}}$ depict either textures on the hat and hair or moderate edges in the background. Notably, few significant triangles lie on the feathers of Lenas hat. This is due to the choice of significant pixels in the AT algorithm, which results in small triangles in this region.



(a) Textural component $\mathbf{I}_t$. (b) Significant triangles $\mathcal{T}_{\text{sig}}$ of $\mathbf{I}_t$.

Figure 5.3: Textural component $\mathbf{I}_t$ of $\mathbf{I}$ given by Lena in figure 5.2. For a better visual recognition, the intensities are shifted and rescaled to the full intensity range $\{0, \dots, 255\}$.

5.3 Basic Compression of Textures

In this section we show the basic reconstruction scheme for the texture component of a given image by demonstrating it on a single significant triangle. As discussed above, we treat the texture image signal as a graph signal on a graph given by vertices defined by the pixels in the triangle. Our aim is to acquire a sparse, quantized representation of the signal in the graph spectral domain that is to be encoded. Figure 5.4 shows the basic outline of this process.

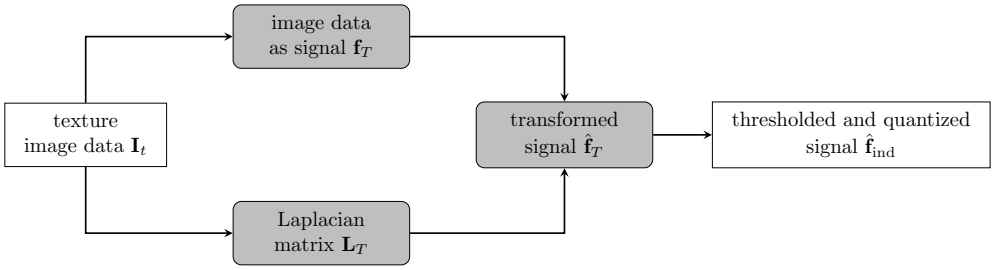


Figure 5.4: Basic reconstruction technique for a significant triangle $T \in \mathcal{T}_{\text{sig}} \subset \mathcal{D}(\mathrm{P}_n)$.

Let $\mathbf{I}_t$ be the texture component of an image $\mathbf{I}$ as in (5.1), where $\widetilde{\mathbf{I}}$ is the AT reconstruction on the significant pixel set P_n. Furthermore, $T \in \mathcal{T}_{\text{sig}} \subset \mathcal{D}(\mathrm{P}_n)$ is a specific significant triangle over which additional information is to be added. In order to treat $\mathbf{I}_t$ as a graph signal, we reevaluate the perception of triangles. A triangle $T \in \mathcal{D}(\mathrm{P}_n)$ is defined in Definition 3.1 as the convex hull of three noncollinear planar points in $\mathrm{Conv}(\mathrm{P}_n)$. To construct a discrete graph, we discretize T by interpreting it as the pixels in the region of the triangle.

Definition 5.2 *Let* $\mathbf{I}$ *be a digital image along with a triangulation* $\mathcal{D}(\mathrm{P}_n)$ *of a significant pixel set* P_n*. Given a triangle* $T \in \mathcal{D}$*, we define its corresponding* discrete triangle *as*

$$T_d = \{p_1, \dots, p_N\} := \mathrm{P} \cap T.$$

From here on out, we do not explicitly distinguish between discrete triangles and regular triangles as in Definition 3.1. It should always be apparent from context which definition is applicable and we simply denote $T \equiv T_d$. We call T either triangular image block or simply image block if the context is unambiguous.

To transfer the texture signal $\mathbf{I}_t$ on T to a graph setting, we construct a connected graph $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T, \mathbf{W}_T\}$ on the set of pixels T and regard $\mathbf{I}_t$ on T as a signal $\mathbf{f}_T$ on $\mathcal{G}_T$.

Naturally $\mathcal{V}_T = T$, where each vertex is associated with a pixel and we may use the terms interchangably. Next, we simply let the value of $\mathbf{f}_T$ at a vertex be the luminance of its associated pixel, i.e. $\mathbf{f}_T(v) = \mathbf{I}_t(p)$ if vertex $v \in \mathcal{V}_T$ is associated with $p \in T$. Note that in this definition the graph signal depends on the order of vertices. For the rest of this thesis we assume that a unique ordering for every triangular image block is given.

The choice of topology, i.e. the edge set $\mathcal{E}_T$, has to be given more care. T is given as a collection of pixels over a regular two-dimensional grid, which forms the basis of the following considerations. Recall that the texture component of an image is the fine scale detail and of oscillating nature, whereas the reconstruction by AT created a global context for each pixel of a triangle. Consequently, we aim to select a graph that captures the small-scale, local variations of intensities in the neighbourhood of a pixel. This is done by connecting pixels based on their physical distance. Two choices utilized in this thesis are the *4-way* and *8-way connectivity graph*.

Definition 5.3 *Let $\mathcal{V} \subset \{(x_1, x_2) \in \mathbb{N} \times \mathbb{N} : 1 \leq x_1 \leq X_1, 1 \leq x_2 \leq X_2\}$ be a finite vertex set on a regular two-dimensinal grid. The* 4-way connectivity graph *is defined as*

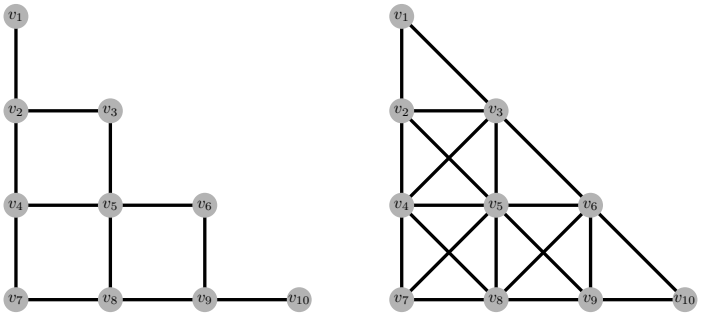
$$\mathcal{E}_4 := \{(v_i, v_j) \in \mathcal{V} \times \mathcal{V} : v_i = v_j \pm (1,0) \vee v_i = v_j \pm (0,1)\},$$

and the 8-way connectivity graph *as*

$$\mathcal{E}_8 := \{(v_i, v_j) \in \mathcal{V} \times \mathcal{V} : v_i = v_j + k \cdot (1,0) + m \cdot (0,1), \text{ where } k, m \in \{-1, 0, 1\}\}.$$

They are constructed from a set of pixels by connecting each vertex to its direct vertical and horizontal neighbours in the 4-way connectivity graph and additionally to its diagonal neighbours in the 8-way connectivity graph. Examples of both the 4-way and 8-way connectivity graph on a set of pixels are depicted in Figure 5.5. In Delaunay triangulations $\mathcal{D}$ the minimum angle of all triangles is maximized. Long, thin triangles are therefore avoided in the construction of $\mathcal{D}$. Thus, we can assume that the 4-way and 8-way connectivity graphs of $T \in \mathcal{T}_{\text{sig}} \subset \mathcal{D}$ are connected. For now we choose the 8-way connectivity graph, since it gives a better context for the localized correlation between pixels.

After constructing the edge set $\mathcal{E}_T$, the corresponding weight for each edge has to be selected in the construction of $\mathbf{W}_T$. We have seen in Chapter 4.3 that the smoothness, and thus the compressibility, of a signal depends on the intrinsic structure of its underlying graph. Especially for textured image blocks the weights in $\mathbf{W}_T$ should be chosen in a fashion that charaterizes the signal, i.e. high weights on edges between similar pixels and low weights between dissimlar



(a) 4-way connectivity graph on $\mathcal{V}$.

(b) 8-way connectivity graph on $\mathcal{V}$.

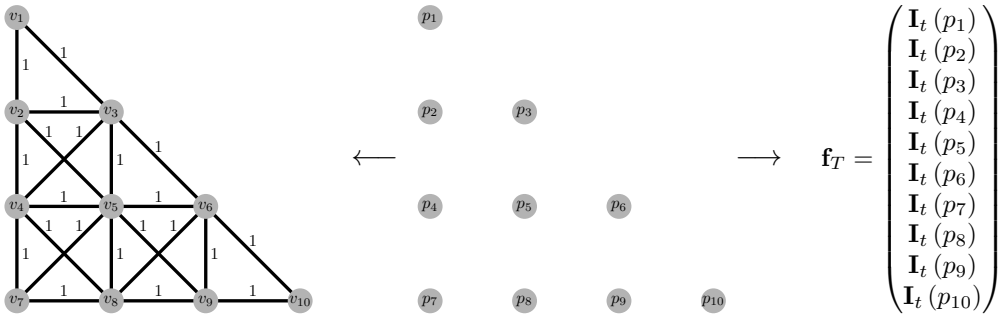
Figure 5.5: Examples of connectivity graphs as given in Definition 5.3 on a triangular image block. The vertex set is given by $\mathcal{V} = \{v_1, \ldots, v_{10}\}$.

vertices. One common way of doing this is via the Gaussian similarity function, see [88], so that

$$W_{i,j} = \exp\left(-\frac{|\mathbf{f}_T(i) - \mathbf{f}_T(j)|^2}{2\alpha^2}\right) \tag{5.2}$$

for all edges $(v_i, v_j) \in \mathcal{E}_T$ given a parameter $\alpha \in \mathbb{R}^+$. But encoding this graph directly is infeasible, as the amount of edges in graphs constructed as above generally exceeds the amount of vertices.

Sections 5.6 and 5.7 aim to construct an efficiently encodable graph for textured image blocks so that $\mathbf{f}_T$ is smooth with regard to it. Therefore, we postpone this discussion to those sections. For smooth blocks without much detail, most of the weights in (5.2) are close to one. Consequently, the standard choice for $\mathbf{W}_T$ is to set the weight of every edge equal to 1, i.e. $W_{i,j} = 1$ for every edge $(i, j) \in \mathcal{E}_T$. Figure 5.6 shows an exemplary construction of the weighted graph $\mathcal{G}_T$ and graph signal $\mathbf{f}_T$ over a triangular image block $T = \{p_1, \ldots, p_{10}\}$ consisting of 10 pixels.



(a) Weighted graph $\mathcal{G}_T$ with constant edge weight 1.

(b) Pixels belonging to triangular image block T

(c) Graph signal $\mathbf{f}_T$

Figure 5.6: Construction of a graph $\mathcal{G}_T$ and signal $\mathbf{f}_T$ from a given image $\mathbf{I}_t$ over a Triangle T comprising 10 pixels.

Having transferred $\mathbf{I}_t$ on T to a graph setting, we aim to acquire a sparse representation of the signal. This is achieved by transforming the signal $\mathbf{f}_T$ to the graph spectral domain via the GFT. To this end, we construct the graph Laplacian $\mathbf{L}_T$ from the weight matrix $\mathbf{W}_T$ and compute its

eigen-decomposition $\mathbf{L}_T = \mathbf{U}_T \mathbf{\Lambda}_T \mathbf{U}_T^T$. We then transform $\mathbf{f}_T$ via

$$\hat{\mathbf{f}}_T = \mathbf{U}_T^T \mathbf{f}_T. \tag{5.3}$$

Following our earlier discussion in Sections 2.1 and 4.4, this decomposes $\mathbf{f}_T$ into its elemental frequencies $\mathbf{u}_k$. The coefficient $\hat{\mathbf{f}}_T(\lambda_k)$ gives the amount of its corresponding frequency $\mathbf{u}_k$ present in $\mathbf{f}_T$ for all $0 \leq k \leq |T| - 1$. Due to the construction of graphs so that the texture signals are smooth with regard to them, most information will be concentrated in few coefficients. This leads to a good reconstruction of $\mathbf{f}_T$ even when only considering a few relevant coefficients.

Some representative elemental frequencies of a smooth graph on an isosceles triangular image block with 1275 vertices are shown in Figure 5.7. Clearly, the fine scale details increase as the frequency increases. While $\mathbf{u}_1$ is constant, $\mathbf{u}_5$ and $\mathbf{u}_{10}$ contain some moderately smooth variations. The density and oscillation of the details in $\mathbf{u}_k$ increase with k, until $\mathbf{u}_{1275}$ consists only of very fine oscillations.

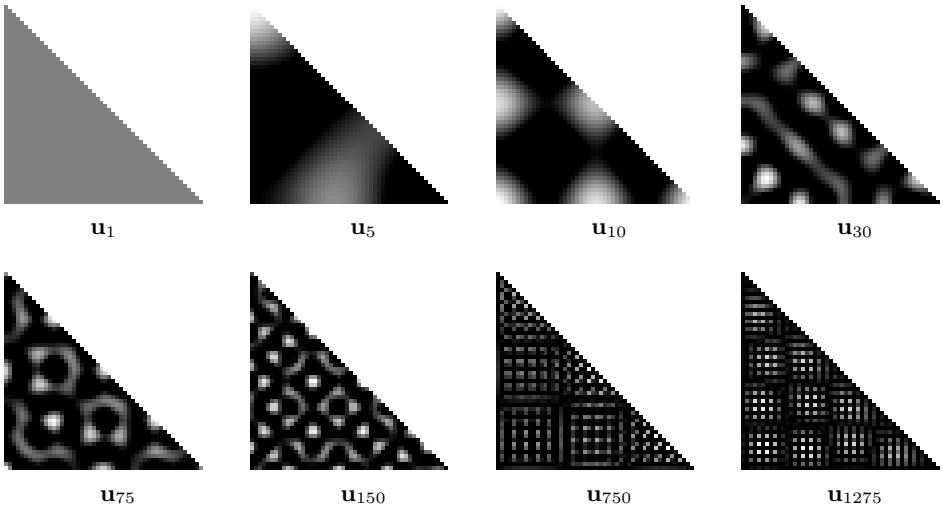


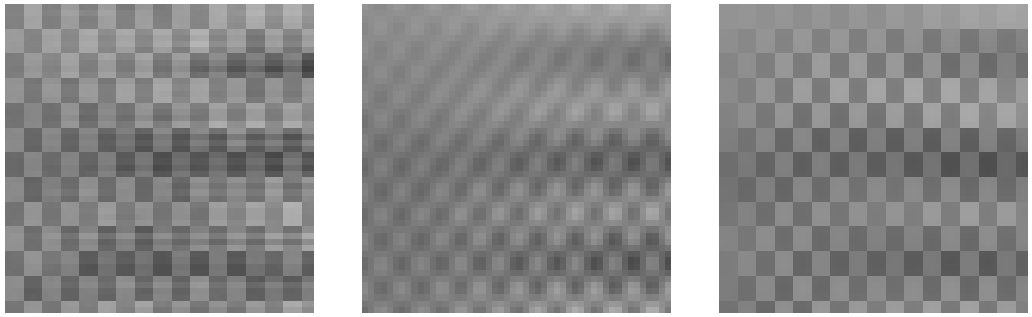
Figure 5.7: A selection of elemental frequencies $\mathbf{u}_k$ of a smooth graph on an isosceles triangular image block comprising 1275 pixels. Each eigenvector is rescaled to show the underlying structure. Note that the fine scale details contained in each eigenvector $\mathbf{u}_k$ increase as its corresponding frequency λ_k increases.

5.4 Signal Smoothness for Textured Blocks

Similarly to the DCT in the JPEG compression scheme in Section 2.3, the GFT in (5.3) is reversible and causes no noteworthy coding gain. It is instead used to enable better quantization as introduced in Section 6.1.4. Due to the choice of standard edge weight 1, smooth triangular image blocks are compressible. They are reconstructed well with just a few low-frequency elemental frequencies displayed in Figure 5.7.

More care has to be given to image blocks that are heavily textured. They contribute more to the degradation of the visual quality of the reconstruction by AT and are subsequently more likely contained in $\mathcal{T}_{\text{sig}}$. We have seen in Section 4.3 that the smoothness of a signal $\mathbf{f}$ depends on the

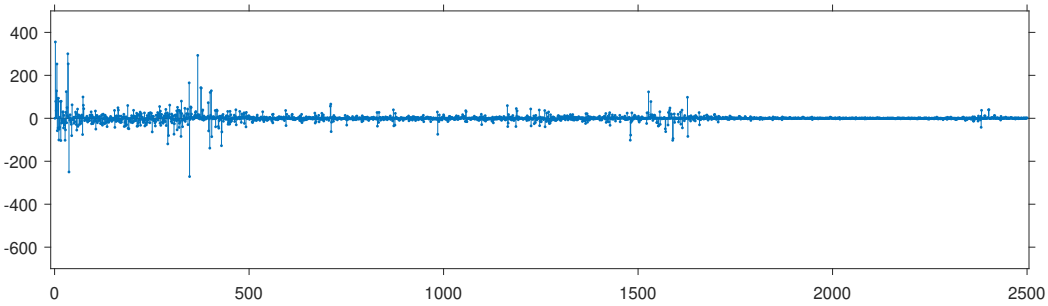
intrinsic structure of the data domain given by $\mathbf{W}$, measured by the graph Laplacian quadratic form $\mathbf{f}^T\mathbf{L}\mathbf{f}$. If the edge weights of a graph given by $\mathbf{L}$ capture the similarities of connected pixels in the signal, $\mathbf{f}$ is smooth and $\mathbf{f}^T\mathbf{L}\mathbf{f}$ will be small. Our aim from here on out is therefore to construct weighted graphs that are efficiently encoded and capture the structure of textures to exploit signal smoothness.



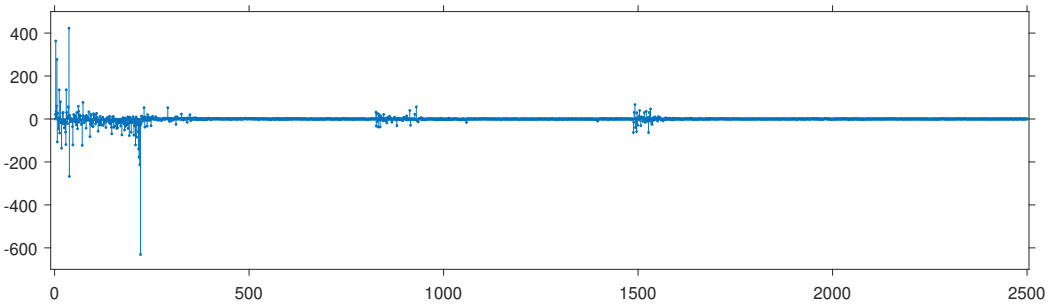
(a) Texture image $\mathbf{I}$.

(b) Reconstruction of $\mathbf{I}$ on $\mathcal{G}_\mathrm{c}$ with SSIM = 0.5536.

(c) Reconstruction of $\mathbf{I}$ on $\mathcal{G}_\mathrm{a}$ with SSIM = 0.8455.



(d) $\hat{\mathbf{f}}_\mathrm{c} = \mathbf{U}_\mathrm{c}^T\mathbf{f}$.



(e) $\hat{\mathbf{f}}_\mathrm{a} = \mathbf{U}_\mathrm{a}^T\mathbf{f}$.

Figure 5.8: Stereotypical texture image block $\mathbf{I}$ transformed via 4-way connectivity graphs. $\mathcal{G}_\mathrm{a}$ is adaptive to $\mathbf{I}$, while $\mathcal{G}_\mathrm{c}$ assigns constant weights to all edges. The reconstructions are based on the ten absolutely largest Fourier coefficients.

The importance of such a graph selection is illustrated in Figure 5.8. It shows a stereotypical texture image $\mathbf{I}$ containing small scale details. 5.8(a) shows the original 50×50 pixel image signal comprising 3×4 pixel substructures. We construct two simple 4-way connectivity graphs, one that adapts to the image and another that does not. In the adaptive graph $\mathcal{G}_\mathrm{a}$, edges between substructures are assigned weight 0.05, while edges within are assigned weight 1. The second graph $\mathcal{G}_\mathrm{c}$ is non-adaptive and simply assigns constant weight 1 to every edge. We let $\mathbf{f}$ be the corresponding graph signal to $\mathbf{I}$ on $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{c}$, albeit shifted to have zero mean so that $\hat{\mathbf{f}}(\lambda_0) = 0$.

The image signal $\mathbf{f}$ is transformed to the graph spectral domain via $\hat{\mathbf{f}}_\mathrm{c} = \mathbf{U}_\mathrm{c}^T\mathbf{f}$ and $\hat{\mathbf{f}}_\mathrm{a} = \mathbf{U}_\mathrm{a}^T\mathbf{f}$. Figures 5.8(d) and 5.8(e) show the Fourier coefficients $\hat{\mathbf{f}}_\mathrm{a}$ and $\hat{\mathbf{f}}_\mathrm{c}$ respectively. Note that the total energy of the signal is spread out much more on the spectrum $\sigma(\mathbf{L}_\mathrm{c})$. On the other hand, the contribution of most frequencies $\lambda_k \in \sigma(\mathbf{L}_\mathrm{a})$ is negligible. More of the total information of $\mathbf{f}$ is contained in just a few low-frequency Fourier coefficients in $\hat{\mathbf{f}}_\mathrm{a}$ compared to $\hat{\mathbf{f}}_\mathrm{c}$.

This is supported when reconstructing $\mathbf{f}$ using only the ten absolutely largest Fourier coefficients of $\hat{\mathbf{f}}_\mathrm{a}$ and $\hat{\mathbf{f}}_\mathrm{c}$. Figure 5.8(c) depicts the reconstruction based on $\mathcal{G}_\mathrm{a}$. It shows a predominantly accurate representation of the substructures, especially the edges between them are distinct. The basic gray scale intensitiy values are reconstructed, even though some deviations exist. On the other hand, the reconstruction based on $\hat{\mathbf{f}}_\mathrm{c}$ in Figure 5.8(b) reproduces only the basic intensitiy gradient. It fails to reconstruct any of the structures contained in the original image. The increased image quality of the reconstruction with an adaptive graph is reflected in the SSIM values. Reconstruction with the non-adaptive graph result in a SSIM value of 0.5536, which is a lot worse than the value for a reconstruction with the adaptive graph with a SSIM value of 0.8455.

Recall from Section 2.1 that a good transformation optimizes the rate-distortion problem given by

$$\min D + \lambda R.$$

Since we aim to design adaptive transformations for suitable image blocks, the total coding cost needs to reflect both the cost of transmitting the signal coefficients, denoted by $R_c(\mathbf{f}, \mathbf{L})$, as well as the cost of the transform description, denoted by $R_\mathbf{L}(\mathbf{L})$. Note that the transform cost is essentially the description cost of $\mathbf{L}$. This gives us the following optimization problem

$$\min_{\substack{\mathbf{L}\in\mathbb{R}^{N\times N}, \\ \mathbf{L} \text{ graph Laplacian}}} D(\mathbf{f}, \mathbf{L}) + \lambda \left(R_c(\mathbf{f}, \mathbf{L}) + R_\mathbf{L}(\mathbf{L})\right).$$

However, if we consider a uniform scalar quantizer with constant step size $\Delta \in \mathbb{R}^+$ for all transform coefficients and assuming high bit rates, i.e. small Δ, the expected value of distortion D is approximated by

$$D = \frac{N\Delta^2}{12},$$

see [40]. N is the amount of coefficients, which remains constant for different orthonormal transforms. For simplicity, we therefore assume that the expected distortion is independent of the transformation for high bit rates. Hence, we only minimize the rate terms.

Definition 5.4 *Given an image signal $\mathbf{f} \in \mathbb{R}^N$, the* optimal GFT *for uniform quantization assuming high bit rates is defined as the minimizer of*

$$\min_{\substack{\mathbf{L}\in\mathbb{R}^{N\times N}, \\ \mathbf{L} \textit{ graph Laplacian}}} R_c(\mathbf{f}, \mathbf{L}) + R_\mathbf{L}(\mathbf{L}).$$

Section 5.7.1 shows how to define proxies for both rates in the case of complex textured blocks.

A different approach to reduce the graph transmission cost $R_{\mathbf{L}}(\mathbf{L})$ is to constrain possible weights to a small, discrete set. To this end, we consider three different possible relations between neighbouring pixels:

Strong correlation: they are strongly correlated if their intensities are very similar. This happens if they are part of a smooth region of an object.

Weak correlation: they are weakly correlated if their intensities are moderately dissimilar. This happens if they are part of a textured region of an object.

No correlation: they are not correlated if their intensities are very dissimilar. This happens if they are part of different geometrical features and bridge a sharp edge.

Recall that the image blocks we are considering are triangles given by the triangulation $\mathcal{D}(\mathrm{P}_n)$ of a set of significant pixels P_n. We have seen in Section 5.2 that they are aligned with sharp edges and describe them well. Hence, we assume that there are no uncorrelated pixels in a specific significant image block. Consequently, we choose edge weights from the discrete set $W_{i,j} \in \{w_\mathrm{o}, 1\}$ for some $w_\mathrm{o} \in (0, 1)$. Neighbouring pixels are connected via an edge with weight 1 if they are strongly correlated and by an edge with weight w_o otherwise. Sections 5.6 and 5.7.2 describe construction methods for graphs with discrete weights, while Section 5.8 shows how to select an optimal edge weight w_o for weakly correlated pixels.

5.5 Classification of Blocks

To distinguish between smooth and textured blocks, we need to estimate the local structure in each image block. Previous approaches have utilized the *structure tensor* (ST) as a tool for this purpose. In [33] it was used as a means to set parameters for an optimization problem, while [75] extracted the dominant principal gradient for ensuing processing. We go one step further and classify all image blocks via the ST and select an appropriate GFT for each class. To this end, we first generalize the ST to an arbitrary domain and apply it to triangular image domains.

For a more detailed overview of the ST we refer to [51].

5.5.1 Block Structure Tensor

The ST summarizes the dominant directions and associated magnitudes of the gradient over a specified window and gives a measure for their coherence. For two-dimensional domains the discrete ST is a 2×2 matrix that represents partial derivatives in a specified neighbourhood of a pixel. As we want to summarize the texturedness of a given image block $T \in \mathcal{T}_\mathrm{sig}$, we modify the conventional definition of the two-dimensional ST. This lets us evaluate the ST of a complete triangular block.

Definition 5.5 *Given an image* $\mathbf{I}$*, the 2-dimensional* block structure tensor *over a pixel set* $T \subset \mathrm{P}$ *is defined as*

$$S_T := \frac{1}{|T|} \begin{pmatrix} \sum\limits_{p \in T} (G_x(p))^2 & \sum\limits_{p \in T} G_x(p) G_y(p) \\ \sum\limits_{p \in T} G_x(p) G_y(p) & \sum\limits_{p \in T} (G_y(p))^2 \end{pmatrix},$$

where $G_x(\cdot)$ *and* $G_y(\cdot)$ *are the horizontal and vertical partial derivatives respectively, evaluated at a pixel.*

The factor $1/|T|$ replaces a weight function $w : T \to \mathbb{R}$ in the conventional ST centered at a single pixel. Each pixel thus has the same impact on the block structure tensor and

$$\sum_{p \in T} w(p) = 1$$

is fulfilled. This normalizes the block structure tensor and ensures comparability between differently sized blocks. From here on out all structure tensors will be on blocks, therefore we refer to them simply as structure tensor or ST.

Performing an eigen-decomposition on the two-dimensional ST S_T, we obtain eigenvalues λ_1 and λ_2, which we order so that $\lambda_1 \geq \lambda_2 \geq 0$. The corresponding eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$ summarize the distribution of the gradient $\nabla G = (G_x, G_y)^T$ in the pixel block T. λ_1 and λ_2 show the strength of the directional intensity change along $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively. $\mathbf{u}_1$, corresponding to the larger eigenvalue λ_1, is the *principal gradient*. The relative difference between λ_1 and λ_2 is quantified by the *coherence*.

Definition 5.6 *Given a block structure tensor* S_T *with eigenvalues* $\lambda_1 \geq \lambda_2 \geq 0$*, where* $\lambda_1 > 0$*, the* coherence c_T *is defined as*

$$c_T := \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2.$$

The coherence c_T indicates the degree of anisotropy of the gradient, with $0 \leq c_T \leq 1$. If the gradient is totally aligned with $\mathbf{u}_1$, then $\lambda_1 > \lambda_2 = 0$ and $c_T = 1$. On the other hand, we have $\lambda_1 = \lambda_2 > 0$ for an isotropic structure where the gradient has no preferred direction and thus $c_T = 0$. Finally, if $\lambda_1 = \lambda_2 = 0$, the intensities in the blocks are constant.

This lets us divide the image blocks into three texture classes:

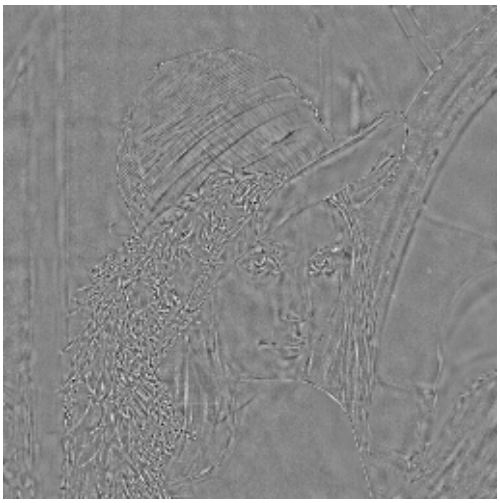
Class 1: smooth blocks with only a small or no detectable gradient in any direction if $\lambda_1 \approx \lambda_2 \approx 0$.

Class 2: blocks with a dominant principal gradient if $\lambda_1 \gg \lambda_2 \approx 0$, i.e. if c_T is large.

Class 3: blocks with no dominant gradient direction and a complex structure if both $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$, i.e. if c_T is small.

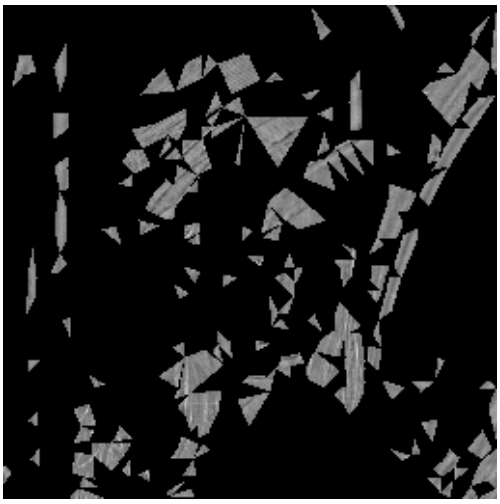
An example of a classification on Lena with a resolution of 256×256 pixels can be seen in Figure 5.9. The textured signal $\mathbf{I}_t$ in 5.9(a) is again derived utilizing the AT reconstruction with $n = 2500$ significant pixels. $\mathcal{D}(\mathrm{P}_{2500})$ is split into the three classes outlined above. Small

image blocks containing less than 20 pixels are not considered in 5.9. If $\lambda_1 < 35$ for a block T, it is considered smooth. Textured blocks with $\lambda_1 \geq 35$ are classified into Class 2 if $c_T > 0.1$, otherwise into Class 3. For a better visual distinguishability, we again shifted and rescaled $\mathbf{I}_t$ to the full intensity space as above. Figure 5.9(b) shows the smooth blocks of $\mathbf{I}_t$ contained in Class 1. Even though there is some detail given, there are no sharp edges or sudden shifts in intensity. In contrast, the remaining two classes show a lot more fine scale detail. But while this detail in figure 5.9(c) is aligned along mostly one direction, it is more complex in figure 5.9(d), as is expected of Class 3.



(a) Textural part $\mathbf{I}_t$ of Lena.

(b) Smooth image blocks, belonging to Class 1.



(c) Textured image blocks with a principal gradient, belonging to Class 2.

(d) Textured image blocks with complex features, belonging to Class 3.

Figure 5.9: Decomposition of the textural component $\mathbf{I}_t$ of an image of Lena on 256×256 pixels, based on $\widetilde{\mathbf{I}}$ with 2500 significant pixels. It is decomposed into the three texture classes introduced above.

5.5.2 Block Structure Tensor on Graphs

In order to construct the ST, we have to sample the partial derivatives $G_x(p)$ and $G_y(p)$ at every pixel $p \in T$. To this end, we construct a 4-way connectivity auxiliary graph $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H}, \mathbf{W}_\mathcal{H})$

with $\mathcal{V}_\mathcal{H} = T$ and constant edge weight $W_{i,j} = 1$ for all $e = (v_i, v_j) \in \mathcal{E}_\mathcal{H}$. Recall from Chapter 4.3 that the graph gradient at a vertex $v_i \in \mathcal{V}_\mathcal{H}$ is given by

$$\nabla_i \mathbf{f} = \left[\left. \frac{\partial \mathbf{f}}{\partial e} \right|_{v_i} \right]_{e \in \mathcal{E}_\mathcal{H} \text{ so that } e=(v_i,\cdot)},$$

where the edge derivative of a single edge $e = (v_i, v_j)$ is defined as

$$\left. \frac{\partial \mathbf{f}}{\partial e} \right|_{v_i} = \sqrt{W_{i,j}}(f(j) - f(i)).$$

Due to the choice of a 4-way connectivity graph, $\nabla_i \mathbf{f}$ comprises a maximum of four possible components along two vertical and two horizontal edges, see Figure 5.10(b). To calculate the directional partial derivatives, we let $v \in T$ be a vertex in the interior of T and v_l, v_r, v_t, $v_b \in T$ the vertices to the left, right, top and bottom of v respectively. They are connected to v via the edges e_l, e_r, e_t, $e_b \in \mathcal{E}_\mathcal{H}$ respectively. Due to the constant edge weight 1, the graph gradient is

$$\nabla_v \mathbf{f} = \begin{pmatrix} \left. \frac{\partial \mathbf{f}}{\partial e_l} \right|_v \\ \left. \frac{\partial \mathbf{f}}{\partial e_r} \right|_v \\ \left. \frac{\partial \mathbf{f}}{\partial e_t} \right|_v \\ \left. \frac{\partial \mathbf{f}}{\partial e_b} \right|_v \end{pmatrix} = \begin{pmatrix} f(v_l) - f(v) \\ f(v_r) - f(v) \\ f(v_t) - f(v) \\ f(v_b) - f(v) \end{pmatrix}. \tag{5.4}$$

We define the directional partial derivatives as

$$\begin{aligned} G_x(v) &:= \left. \frac{\partial \mathbf{f}}{\partial e_r} \right|_v + \left(- \left. \frac{\partial \mathbf{f}}{\partial e_l} \right|_v \right) = f(v_r) - f(v_l), \\ G_y(v) &:= \left. \frac{\partial \mathbf{f}}{\partial e_b} \right|_v + \left(- \left. \frac{\partial \mathbf{f}}{\partial e_t} \right|_v \right) = f(v_b) - f(v_t). \end{aligned} \tag{5.5}$$

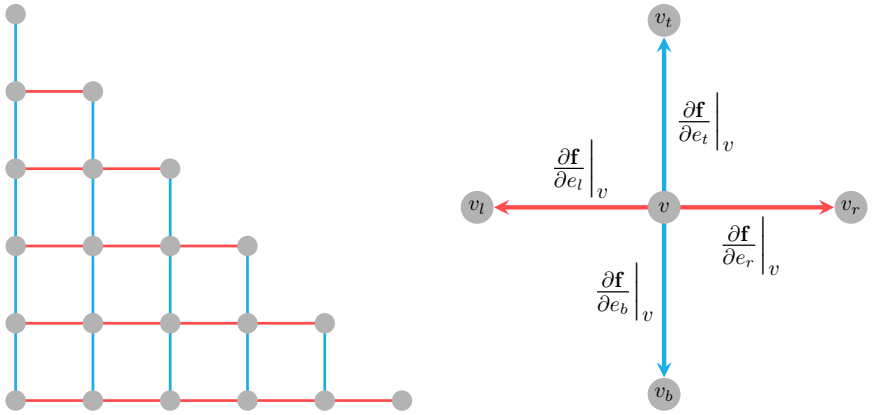
Note that the derivatives in (5.4) are oriented away from vertex v. To orient them uniformly from left to right and from top to bottom, we reverse the direction of $\left. \frac{\partial \mathbf{f}}{\partial e_l} \right|_v$ and $\left. \frac{\partial \mathbf{f}}{\partial e_t} \right|_v$ in (5.5). This evaluation of G_x and G_y is similar to the image gradient widely used in edge detection, see for example [52], but constrained to triangular blocks defined on graphs.

For pixels not in the interior of T we choose a replicating boundary condition. That is, pixels not in T are assumed to be of equal value as v. If a direct neighbour v_* as above does not exist for $v \in T$, we set $\mathbf{f}(v_*) = \mathbf{f}(v)$ in (5.4). This implies $\left. \frac{\partial \mathbf{f}}{\partial e_*} \right|_v = 0$ for $e_* = (v, v_*) \notin \mathcal{E}_\mathcal{H}$ in (5.5).

5.6 Blocks with Dominant Principal Gradient

We first turn to the graph construction on image blocks in Class 2 with a dominant principal gradient. The approach here is similar to [76], where image blocks are clustered into K groups on which a graph with adaptive edges and edge weights is designed. We reduce the overhead cost by considering only four groups with fixed graph templates and select the edge weights to be in the set $\{w_\mathrm{o}, 1\}$.

As introduced in Section 5.5, a texture block $T \in \mathcal{D}(\mathrm{P}_n)$ belongs to Class 2 if $\lambda_1 > \lambda_2 \approx 0$



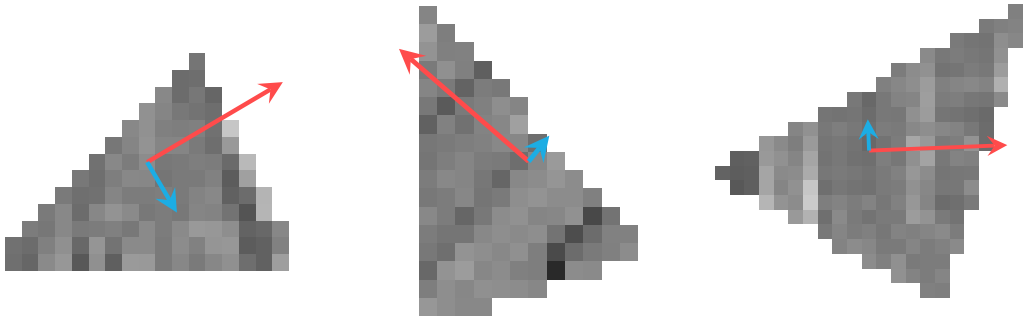
(a) Image block T with $\mathcal{E}_{\mathcal{H}}$.

(b) The four neighbours of v in the interior of T.

Figure 5.10: Computation of partial derivatives G_x and G_y on an image block T. Red edges contribute to the computation of G_x and blue edges to the computation of G_y.

and c_T is large. The ST eigenvalues λ_1 and λ_2 reflect the distribution of the gradient in a block. Thus, the eigenvector $\mathbf{u}_1$, corresponding to λ_1, represents the principal gradient, i.e. the vector maximally aligned with the block gradient. An adaptive GFT can exploit this information for coding gain.

Given a triangular image block T belonging to Class 2 along with its principal gradient $\mathbf{u}_1$, we construct a graph $\mathcal{G}_T$ that is connected and represents the structure of the signal. Edges connecting pixels with similar intensity values should have standard weight 1 and edges between pixels with dissimilar values should have weight w_o. Some examples of Class 2 image blocks from Lena in Figure 5.3 are shown in Figure 5.11.



(a) Image block in Group 4 with $c_T = 0.2755$.

(b) Image block in Group 2 with $c_T = 0.3094$.

(c) Image block in Group 3 with $c_T = 0.3693$.

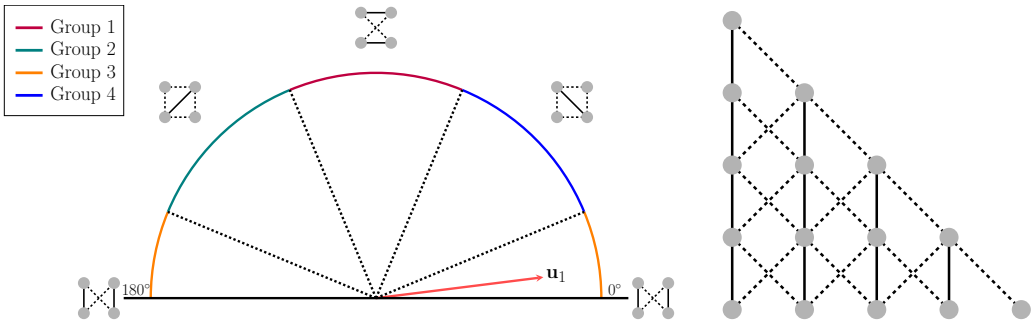
Figure 5.11: Examples of image blocks in Class 2, along with the eigenvectors of the ST and their coherence. The principal gradient $\mathbf{u}_1$ is shown in red and $\mathbf{u}_2$ in blue. We remark that the blocks are not to scale.

As the gradient is aligned with $\mathbf{u}_1$ in T, we expect large intensity differences in the direction of $\mathbf{u}_1$. On the other hand, similar intensity values should be predominant in the direction orthogonal to $\mathbf{u}_1$. Consequently, we assign weight w_o to edges that align somewhat with $\mathbf{u}_1$ and weight 1 to edges that are mostly orthogonal to $\mathbf{u}_1$.

To this end, we distinguish between four groups of blocks based on the direction of $\mathbf{u}_1$. We assign each group a suitable graph template to subsequently construct $\mathcal{G}_T$. A block with dominant principal gradient is assigned a group by selecting the orientation $q_{\mathbf{u}_1} \in \{0°, 45°, 90°, 135°\}$ that is most orthogonal to $\mathbf{u}_1$. This can be understood as a quantization of gradient directions. Based on the predominant gradient direction, we modify the regular 8-way connectivity graph template. The weight of edges orthogonal to $q_{\mathbf{u}_1}$ is set to 0, edges aligned with $q_{\mathbf{u}_1}$ are assigned weight 1 and all other edge weights are set to w_o:

- Group 1: $\mathbf{u}_1$ is most orthogonal to $q_{\mathbf{u}_1} = 0°$ and T consists mostly of horizontal structures. Vertical edges are assigned weight 0, diagonal edges weight w_o and horizontal edges weight 1.
- Group 2: $\mathbf{u}_1$ is most orthogonal to $q_{\mathbf{u}_1} = 45°$ and T consists mostly of structures from bottom-left to top-right. Diagonal edges along this orientation are assigned weight 1, horizontal and vertical edges weight w_o and all other edges weight 0.
- Group 3: $\mathbf{u}_1$ is most orthogonal to $q_{\mathbf{u}_1} = 90°$ and T consists mostly of vertical structures. Horizontal edges are assigned weight 0, diagonal edges weight w_o and vertical edges weight 1.
- Group 4: $\mathbf{u}_1$ is most orthogonal to $q_{\mathbf{u}_1} = 135°$ and T consists mostly of structures from top-left to bottom-right. Diagonal edges along this orientation are assigned weight 1, horizontal and vertical edges weight w_o and all other edges weight 0.

The resulting groups and group templates are shown in figure 5.12(a). For example, a block with principal gradient $\mathbf{u}_1$ as in 5.12(a) belongs to Group 3 since it is most orthogonal to $q_{\mathbf{u}_1} = 90°$. Depending on the principal gradient $\mathbf{u}_1$, the graph template of the associated region is applied to the block. Basis for this is the standard 8-way connectivity graph, where each vertex is adjacent to its eight neigbours. The weights are then adjusted in accordance to the selected group template, where assigning weight 0 removes the edge from $\mathcal{E}_T$. Figure 5.12(b) shows how to apply the graph template for a block with $q_{\mathbf{u}_1} = 90°$.



(a) Graph templates corresponding to groups 1 to 4.

(b) Graph constructed for T in Group 3.

Figure 5.12: Classification of image blocks with a dominant principal gradient $\mathbf{u}_1$ into four groups, along with the corresponding graph templates. Continuous lines represent edges with weight 1 and dashed lines edges with weight w_o.

The influence of the structure introduced by the graph templates can be seen in Figure 5.13, where we take a model trianglular image block T and display some elemental frequencies for the different groups. T consists of 1275 pixels arrayed in an isosceles triangle. We consider the elemental frequencies $\mathbf{u}_5$, $\mathbf{u}_{10}$, $\mathbf{u}_{30}$, $\mathbf{u}_{750}$ and $\mathbf{u}_{1275}$ of graphs $\mathcal{G}_k$, each constructed in accordance to the graph template of Group k, for $1 \leq k \leq 4$. The weight between weakly correlated pixels is set to $w_{\mathrm{o}} = 0.25$.

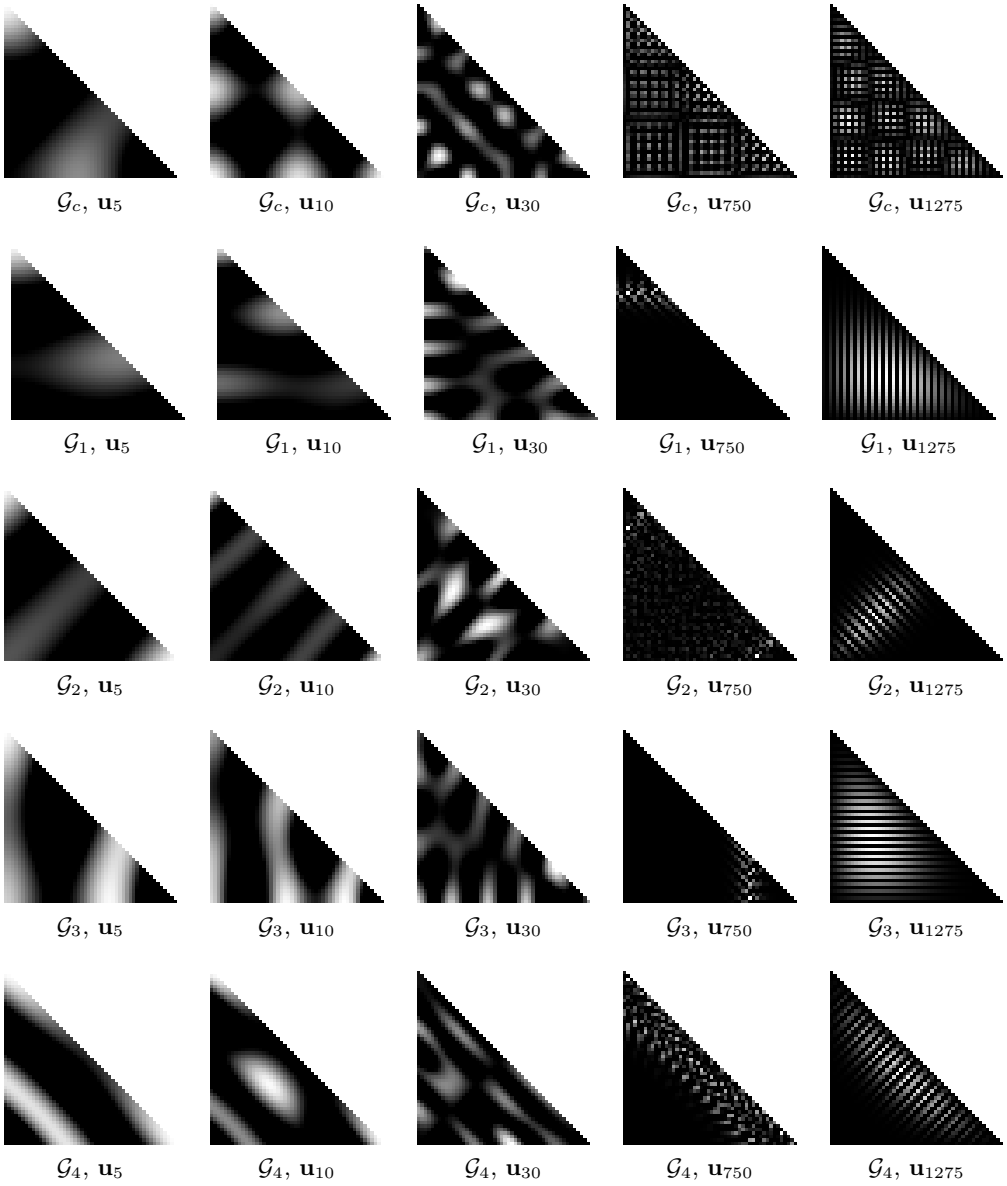


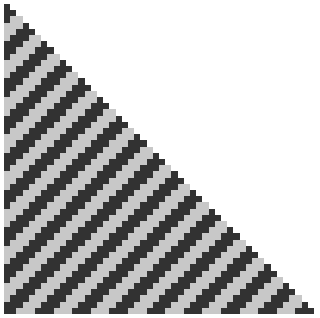
Figure 5.13: Selection of elemental frequencies $\mathbf{u}_k$ of an isosceles triangle with 1275 pixels on $\mathcal{G}_c$ and $\mathcal{G}_k$, $1 \leq k \leq 4$, rescaled to the whole image range $[0, 255]$.

For comparison, we show the elemental frequencies of a 8-way connectivity graph $\mathcal{G}_c$ with constant edge weight 1 in the first row, as seen in Figure 5.7. Note that these elemental frequencies are mainly isotropic and do not have a clear directional texture. This changes when we consider any of the four directional groups described above.

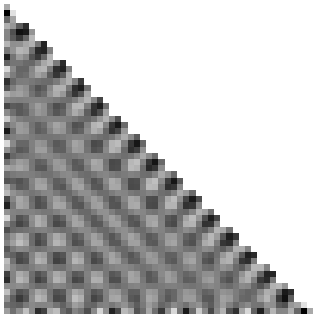
Although the basic textures are similar in the three smaller frequencies, they still show a clear alignment in the direction established by the graph template. As the frequency increases,

we observe larger deviations to the textures given by the constant graph. They are still aligned along the established direction but more localized. This localization is an additional benefit of this scheme for capturing textures. Eigenvectors $\mathbf{u}_k$ tend to have a similar structure for similar values of k, especially in large frequencies. Typically, this results in similar structures with different localizations. Hence, localized features in a textured block are represented well by single elemental frequencies. Finally, $\mathbf{u}_{1275}$ shows the eigenvectors corresponding to the highest frequency. As they are the coarsest eigenvectors of the underlying graphs, we observe that while the larger scale is still aligned with the template, large differences occur along edges orthogonal to $q_{\mathbf{u}1}$.

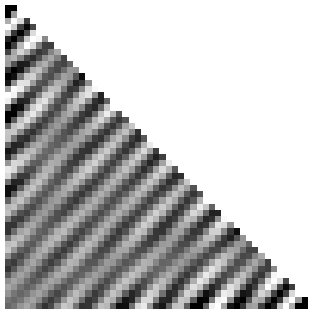
To summarize the previous discussion, we note that the alignment with the template is present among all frequencies, albeit more pronounced in lower frequencies.



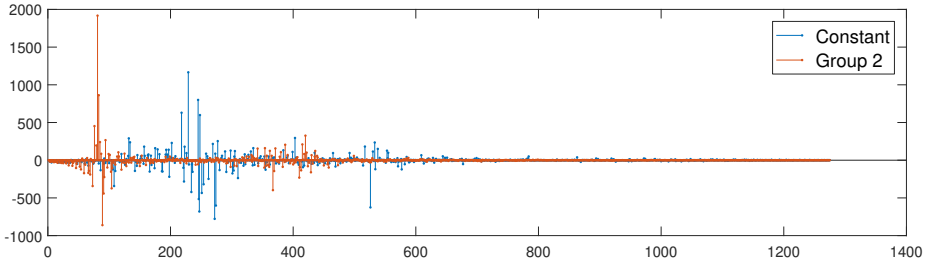
(a) Original signal $\mathbf{f}$ on T.



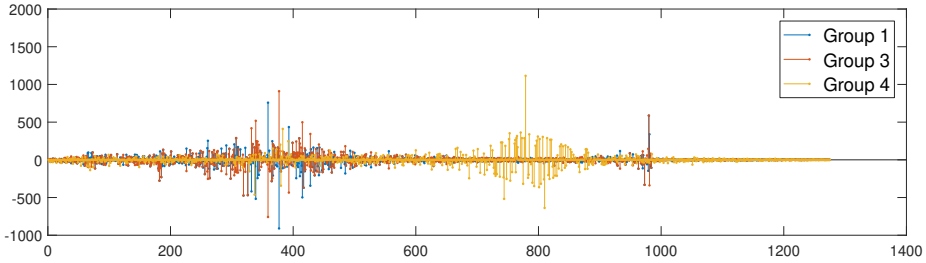
(b) Reconstruction from $\hat{\mathbf{f}}_c$. SSIM = 0.7044.



(c) Reconstruction from $\hat{\mathbf{f}}_2$. SSIM = 0.8826.



(d) $\hat{\mathbf{f}}_c$ and $\hat{\mathbf{f}}_2$.



(e) $\hat{\mathbf{f}}_1$, $\hat{\mathbf{f}}_3$ and $\hat{\mathbf{f}}_4$.

Figure 5.14: Fourier coefficients of a stereotypical signal $\mathbf{f}$ with a dominant principal gradient orthogonal to $q_{\mathbf{u}_1} = 45°$. Additionally, reconstructions with two Fourier coefficients of $\mathbf{f}$ over $\mathcal{G}_c$ and $\mathcal{G}_2$ are shown. The reconstruction from $\hat{\mathbf{f}}_2$ is clearly superior.

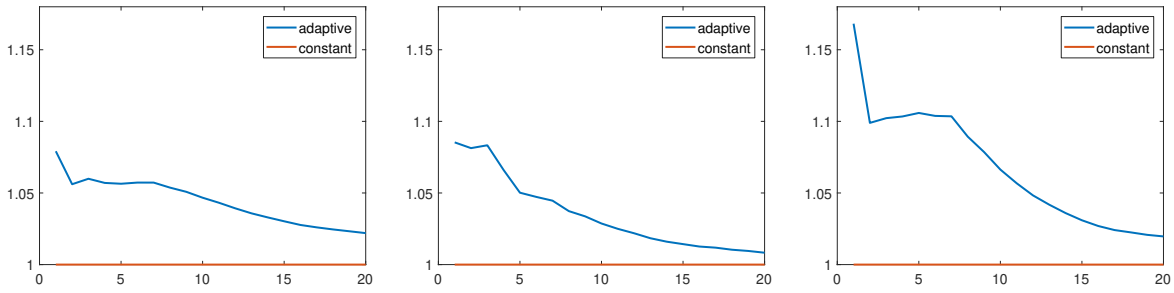
In order to support the theory above, we compare the performance of a constant graph and different adaptive graphs on a stereotypical image block in Group 2. To this end, we let T be

a triangular image block with an image signal $\mathbf{f}$ belonging to Group 2, see Figure 5.14. T is again an isosceles triangle consisting of 1275 pixels, with $\mathbf{f}$ given on T containing structures from bottom-left to top-right, depicted in Figure 5.14(a). We construct a smooth graph $\mathcal{G}_c$ and adaptive graphs $\mathcal{G}_k$ for $1 \leq k \leq 4$ according to the graph template of each respective group. During the construction of the adaptive graphs, the edge weight for weak transitions is set to $w_o = 0.25$. We compute their graph Fourier coefficients $\hat{\mathbf{f}}_c$, $\hat{\mathbf{f}}_k$ for $1 \leq k \leq 4$ as in (5.3).

Figure 5.14(d) compares $\hat{\mathbf{f}}_c$ (blue signal) with $\hat{\mathbf{f}}_2$ (red signal). Since more energy in $\hat{\mathbf{f}}_2$ is compacted in the lower frequencies along with larger single coefficients, it is clear that $\mathbf{f}$ is smoother in regard to $\mathcal{G}_2$ than to $\mathcal{G}_c$. This is confirmed by a reconstruction with the two largest Fourier coefficients in $\hat{\mathbf{f}}_c$ and $\hat{\mathbf{f}}_s$, see Figure 5.14(b) and 5.14(c) respectively. The reconstruction on $\mathcal{G}_2$ manages to reconstruct the basic diagonal structure, with a SSIM-value of 0.8826. This is a lot better than a reconstruction on $\mathcal{G}_c$ that fails to do so completely, which is reflected in a lower SSIM-value of 0.7044.

Figure 5.14(e) displays the Fourier coefficients of the other groups. The coefficients $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_3$ are very similar, which was to be expected as they are both semi-aligned with the regular main structures. Both are evidently coarser than $\hat{\mathbf{f}}_2$, since their main energy is concentrated in higher frequencies than even $\hat{\mathbf{f}}_c$, with a peak in very high frequencies. As expected, $\hat{\mathbf{f}}_4$ is the coarsest, since its underlying template assumes structures orthogonal to those present in $\mathbf{f}$.

This subjective impression is supported by the values of the graph Laplacian quadratic forms. Each adaptive graph has a comparable amount of edges with weight 1 and edges with weight w_o, thus the values of $\mathbf{f}^T\mathbf{L}_k\mathbf{f}$ for $1 \leq k \leq 4$ are comparable. $\mathcal{G}_2$ has the lowest value with $\mathbf{f}^T\mathbf{L}_2\mathbf{f} \approx 3.67 \times 10^6$, which again indicates the smoothness of $\mathbf{f}$ in regard to $\mathcal{G}_2$. The values for $\mathcal{G}_1$ and $\mathcal{G}_3$ are equal, $\mathbf{f}^T\mathbf{L}_1\mathbf{f} = \mathbf{f}^T\mathbf{L}_3\mathbf{f} \approx 1.29 \times 10^7$, which is almost four times the value of $\mathcal{G}_1$. $\mathcal{G}_4$ again performs the worst, with $\mathbf{f}^T\mathbf{L}_4\mathbf{f} \approx 2.2 \times 10^7$.



(a) $\lambda_1 \geq 35$ and $c_T \geq 0.2$. (b) $\lambda_1 \geq 100$ and $c_T \geq 0.2$. (c) $\lambda_1 \geq 35$ and $c_T \geq 0.4$.

Figure 5.15: Percentage of energy contained in k largest Fourier coefficients on blocks in Class 2. Only those blocks are taken into account that fulfill the constraints.

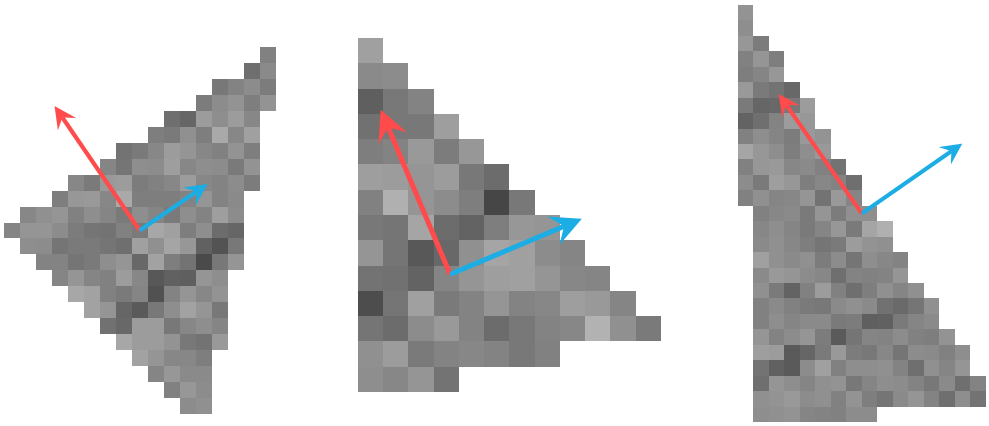
To demonstrate the performance on a natural image, we again consider Lena. Here we reconstruct $\mathbf{I}$ with $n = 4000$ significant pixels, as this is more reasonable in a practical application. As above, we compare the performance of adaptive graphs constructed with the graph template of their respective groups with that of a constant graph $\mathcal{G}_c$. Figure 5.15 shows the average cumulative squared energy of the first k largest Fourier coefficients on image blocks T in Class 2. The cumulative squared energy is displayed in relation to the energy contained in $\hat{\mathbf{f}}_c$ for different

choices of cutoff-values for the coherence c_T for λ_1.

We observe that the GFT with adaptive graphs accumulates more energy than the corresponding constant graph, especially when considering few Fourier coefficients. Increasing the cutoff-value for λ_1 leads to more energy compacted in the same amount of Fourier coefficients, since this signifies more structured texture patterns. The accumulated energy increases considerably more if we increase the cutoff-value for the coherence. This is expected, since this signifies more alignment in one direction, which greatly benefits the compression via adaptive graphs based on $q_{\mathbf{u}_1}$. We remark that the improved visual quality due to emphasized texture patterns is not reflected in this measure.

5.7 Blocks with complex features

In this section we turn to the graph construction on image blocks $T \in \mathcal{D}(\mathrm{P}_n)$ in Class 3. Unlike in the last section, T contains complex features and the gradient has no predominant direction. It is therefore not possible to summarize the textures on T by a simple directional vector. Examples of image blocks in Class 3 are shown in figure 5.16.



(a) Complex image block with $c_T = 0.0860$.

(b) Complex image block with $c_T = 0.0124$.

(c) Complex image block with $c_T = 0.0196$.

Figure 5.16: Examples of image blocks in Class 2, along with the eigenvectors of the ST and their coherence. The eigenvector $\mathbf{u}_1$ is shown in red and $\mathbf{u}_2$ in blue. We remark that the blocks are not to scale.

We approach the problem with two different models. The first approach follows along the lines of [33]. It defines an optimization problem that minimizes the rate-distortion cost as in Definition 5.4. In the second approach we choose weights in a block from a discrete set $\{w_o, 1\}$. But here we explicitly assume low bit rates and construct the graph accordingly. During the encoding we evaluate which mode is more suitable to encode the given signal, see Section 6.1.3.

To compress the signal efficiently, we reduce the transmission cost of the transform. Recall from Section 4.4 that given a graph $\mathcal{G}$, its graph Laplacian $\mathbf{L}$ is constructed from its incidence matrix $\mathbf{J}$ and edge weight vector $\mathbf{w}$ via

$$\mathbf{L} = \mathbf{J}^T \mathrm{diag}(\mathbf{w})\mathbf{J}. \tag{5.6}$$

In order to reduce the graph transmission cost, we fix the incidence matrix $\mathbf{J}$ and thus the topology of the graph. As a graph with fixed incidence matrix is represented by $\mathbf{w}$, the graph transmission cost grows proportionally with the amount of edges $M = |\mathcal{E}|$ in $\mathcal{G}$. Therefore, we apply the simpler 4-way connectivity template for the topology of $\mathcal{G}$, since it contains only about half as many edges as the 8-way connectivity graph.

Having fixed the topology of $\mathcal{G}$, we compute $\mathbf{L}$ simply from $\mathbf{w}$. To reduce the transmission cost further, we interpret $\mathbf{w}$ in both approaches as a graph signal on the unweighted line graph $\mathcal{G}_\ell$ of $\mathcal{G}$, see Section 4.2. Note that the line graph is uniquely determined by the pixels in T and the topology defined above. We weigh $\mathcal{G}_\ell$ by assigning each link in $\mathcal{E}_\ell$ weight 1. Subsequently, we construct the graph Laplacian $\mathbf{L}_\ell \in \mathbb{R}^{M\times M}$ of $\mathcal{G}_\ell$ and compute its eigen-decomposition , so that $\mathbf{L}_\ell = \mathbf{U}_\ell \mathbf{\Lambda}_\ell \mathbf{U}_\ell^T$. Since $\mathbf{w} \equiv \mathbf{w}_\ell$ is a signal on $\mathcal{G}_\ell$, we compute its GFT. It transforms $\mathbf{w}$ from the vertex domain to the graph spectral domain via

$$\hat{\mathbf{w}} = \mathbf{U}_\ell^T \mathbf{w}.$$

Thus, we can represent $\mathbf{L}$ equivalently by $\hat{\mathbf{w}}$. We again aim to select $\mathbf{w}$ so that it is reconstructable by few Fourier coefficients in $\hat{\mathbf{w}}$. What remains to be done is to construct the weight vector $\mathbf{w} \in (0,1]^M$ which additionally reflects the similarities between neighbouring pixels.

5.7.1 Optimization of the Graph Transform

Recall from Definition 5.4 that the optimal GFT minimizes the total cost for the graph description of a signal $\mathbf{f}$ on an image block

$$\min_{\substack{\mathbf{L}\in\mathbb{R}^{N\times N}, \\ \mathbf{L} \text{ graph Laplacian}}} R_c(\mathbf{f},\mathbf{L}) + R_{\mathbf{L}}(\mathbf{L}).$$

We recast this minimization problem by constructing $\mathbf{L}$ as in (5.6), so that R_c and $R_{\mathbf{L}}$ depend on $\mathbf{w}$

$$\min_{\mathbf{w}\in\mathbb{R}^M} R_c(\mathbf{f},\mathbf{w}) + R_{\mathbf{L}}(\mathbf{w}). \tag{5.7}$$

To solve this problem, we follow along the lines of [45] and [33], starting by introducing rate proxies for $R_c(\mathbf{f},\mathbf{w})$ and $R_T(\mathbf{w})$.

As per our previous discussion in Section 5.4, we evaluate the coefficient description cost R_c of a transform based on the smoothness of the image signal with regard to the graph. We have already discussed how the graph Laplacian quadratic form is a measure for the smoothness of signal. Consequently, we use the proxy proposed in [45] and [56], namely

$$\begin{aligned} R_c(\mathbf{f}, \mathbf{w}) &= \mathbf{f}^T \mathbf{J}^T \mathrm{diag}(\mathbf{w}) \mathbf{J} \mathbf{f} \\ &= \mathbf{f}^T \mathbf{L} \mathbf{f} \\ &= \mathbf{f}^T \left(\sum_{k=0}^{N-1} \lambda_k \mathbf{u}_k \mathbf{u}_k^T \right) \mathbf{f} \\ &= \sum_{k=0}^{N-1} \lambda_k (\mathbf{f}^T \mathbf{u}_k)(\mathbf{u}_k^T \mathbf{f}) \\ &= \sum_{k=0}^{N-1} \lambda_k \left(\hat{\mathbf{f}}(\lambda_k) \right)^2, \end{aligned} \tag{5.8}$$

where $\mathbf{u}_k$ and λ_k, for $0 \leq k \leq N-1$, are the k^{th} eigenvector and eigenvalue of $\mathbf{L}$ and $\hat{\mathbf{f}}(\lambda_k)$ is the k^{th} Fourier coefficient. Equation (5.8) shows that $\mathbf{f}^T \mathbf{L} \mathbf{f}$ is an eigenvalue-weighted sum of squared transform coefficients. As such, the DC coefficients corresponding to eigenvalue $\lambda_0 = 0$ are not taken in account. But since the topology is fixed and we assume to have a connected graph, we can ignore this cost.

For the transform description cost $R_{\mathbf{L}}(\mathbf{w})$, we treat $\mathbf{w}$ as a signal on the line graph. To this end, we transform $\mathbf{w}$ to the graph spectral domain and evaluate the coding cost of $\hat{\mathbf{w}}$. Even though we treat both $\mathbf{f}$ and $\mathbf{w}$ as graph signals, we do not use the same coding cost approximation for $\hat{\mathbf{w}}$ that we used for $\hat{\mathbf{f}}$. The purpose of the approximation in (5.8) is to obtain a graph so that $\mathbf{f}$ is smooth with regard to it. But we know that the current image block has a complex structure, so $\mathbf{w}$ will not be smooth with regard to the constant graph $\mathcal{G}_\ell$. Instead, as proposed in [33], we measure the transmission cost based on the sparsity of $\hat{\mathbf{w}}$. In [62] it was shown that the cost of coding a vector is proportional to the number of non-zero coefficients. Thus, the proxy for the transform description cost has to quantify the sparsity of $\hat{\mathbf{w}}$. Unfortunately, minimizing the sparsity of $\hat{\mathbf{w}}$ is NP-hard, see [32], and therefore infeasible. Similarly to compressive sensing, we instead approximate the sparsity of $\hat{\mathbf{w}}$ by the ℓ^1-norm, so that

$$R_{\mathbf{L}}(\mathbf{w}) := \|\hat{\mathbf{w}}\|_{\ell^1} = \left\| \mathbf{U}_\ell^T \mathbf{w} \right\|_{\ell^1}. \tag{5.9}$$

Before combining the proxies of the rate optimization in (5.7), we limit possible choices of $\mathbf{w}$. Recall that our construction of $\mathbf{W}$ in Chapter 4 only permitted positive weights. Furthermore, for a controlled quantization, we set the maximum weight to 1, so that $\mathbf{0} < \mathbf{w} \leq \mathbf{1}$, where $\mathbf{0}$ and $\mathbf{1}$ denote the constant 0 and 1 vector respectively. In fact, most common weighting functions (such as the Gaussian weighting function introduced in Section 5.2) select weights in the range $(0, 1]$, see [39].

Additionally, we penalize low weights by adding a logarithmic term, as is often done in graph

learning problems, see [54]. This is necessary because the smoothness measure $\mathbf{f}^T\mathbf{L}\mathbf{f}$ does not reflect the relation between weights. Choosing a lower weight for an edge is always better when measuring the smoothness by $\mathbf{f}^T\mathbf{L}\mathbf{f}$. Thus, a low weight is selected between dissimilar values, but also between similar values. The relation between the weights is completely disregarded. By adding the logarithmic term, lower weights are only selected when the corresponding vertices have dissimilar values, which coincides with our understanding of smoothness. The logarithmic term has the additional benefit that it guarantees $\mathbf{w} > 0$, which lets us omit the lower bound $\mathbf{w} > \mathbf{0}$.

By combining (5.7), (5.8) and (5.9) and taking the previous considerations into account, we obtain the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathbb{R}^M} \mathbf{f}^T\mathbf{L}\mathbf{f} + \alpha \left\| \mathbf{U}_\ell^T \mathbf{w} \right\|_{\ell^1} - \beta \mathbf{1}^T \log(\mathbf{w}), \\ &\text{s.t. } \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{5.10}$$

To solve this problem, recall that $\mathbf{f}^T\mathbf{L}\mathbf{f}$ can be rewritten as the weighted sum of the squared differences along edges. This lets us rewrite the first term in (5.10).

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{(i,j)\in\mathcal{E}} W_{i,j}(\mathbf{f}(j) - \mathbf{f}(i))^2 = \mathbf{s}^T\mathbf{w},$$

where $\mathbf{s} \in \mathbb{R}^M$ is the vector that contains the squared intensity differences along each edge, i.e. $\mathbf{s}(e) = (\mathbf{f}(i) - \mathbf{f}(j))^2$ for an edge $e = (i, j)$. We remark that $\mathbf{s}$ only depends on $\mathbf{J}$ and $\mathbf{f}$, not on $\mathbf{w}$. Finally, we rewrite the problem in (5.10) as

$$\begin{aligned} &\min_{\mathbf{w}\in\mathbb{R}^M} \mathbf{s}^T\mathbf{w} + \alpha \left\| \mathbf{U}_\ell^T \mathbf{w} \right\|_{\ell^1} - \beta \mathbf{1}^T \log(\mathbf{w}), \\ &\text{s.t. } \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{5.11}$$

This is a convex constrained minimization problem with respect to $\mathbf{w}$ and solved efficiently by interior-point methods, see [5].

5.7.2 Thresholding Edge Differences

While the optimization in the previous section results in an optimal GFT for high bit rates, this does not necessarily transfer to low bit rates. To account for this, we propose an alternative approach to construct the edge weight vector $\mathbf{w}$. According to the discussion in Section 5.4, we select the possible edge weight for each edge $e = (i, j) \in \mathcal{E}$ from a discrete set $W_{i,j} \in \{w_\mathrm{o}, 1\}$ for some $w_\mathrm{o} \in (0, 1)$. This lets us encode each edge weight as a 1-bit signal. While this already reduces the graph description cost significantly, we can improve this by transforming $\mathbf{w}$ to the graph spectral domain. Each elemental frequency $\mathbf{u}_{\ell,k}$ defines a 1-bit signal when considering the sign of each element. When reconstructing $\mathbf{w}$ from $\hat{\mathbf{w}}$, the exact values are not important, only the signs are. This leads to a more forgiving reconstruction scheme even with few Fourier coefficients.

The approach has to fulfill two criteria: (i) distinguish between weak and strong correlation and (ii) emphasize the importance of correlation. Criterium (i) is obvious, since assigning an edge weak or regular weight defines the weight vector $\mathbf{w}$. Criterium (ii) is important, since a perfect reconstruction is almost impossible for low bit rates. Additionally, edge differences have a wide variety of values. Large dissimilarities along edges are very important for the reconstruction and should be reconstructed more accurately. This approach is designed to emphasize the right choice for very similar or dissimilar values at the cost of misclassifying less important edges.

To distinguish between weakly and strongly correlated pixels, we threshold the intensity differences along the edges of $\mathcal{G}$. If the intensity difference along an edge $e = (v_i, v_j)$ is large, we select a low weight $W_{i,j} = w_\mathrm{o}$, otherwise regular weight $W_{i,j} = 1$.

Given an image signal $\mathbf{f}$ on a triangular image block in Class 3 and the line graph $\mathcal{G}_\ell$ of this block, we collect the intensity differences of neighbouring pixels in a vector $\mathbf{d} \in \mathbb{R}^M$, i.e.

$$\mathbf{d}(e) = \mathbf{f}(i) - \mathbf{f}(j)$$

for every edge $e = (v_i, v_j) \in \mathcal{E}$. We set the weight for every edge based on $\mathbf{d}$ and a threshold $\gamma \in \mathbb{R}^+$. The threshold is chosen based on the mean and variance of the image signal, see Section 6.1.3. Once γ is fixed, the weight of an edge $e = (v_i, v_j) \in \mathcal{E}$ is chosen via

$$W_{i,j} = \mathbf{w}(e) = \begin{cases} w_\mathrm{o} & \text{if } |\mathbf{d}(e)| > \gamma, \\ 1 & \text{if } |\mathbf{d}(e)| \leq \gamma. \end{cases} \tag{5.12}$$

We translate the intensity differences by γ, i.e.

$$\mathbf{d}_\gamma = |\mathbf{d}| - \gamma \mathbf{1}.$$

This lets us reframe the classification in (5.12) as

$$W_{i,j} = \mathbf{w}(e) = \begin{cases} w_\mathrm{o} & \text{if } \mathbf{d}_\gamma(e) > 0, \\ 1 & \text{if } \mathbf{d}_\gamma(e) \leq 0. \end{cases} \tag{5.13}$$

Note that for a given incidence matrix $\mathbf{J}$ and weight w_o, the graph Laplacian $\mathbf{L}$ can be reconstructed directly from $\mathbf{d}_\gamma$, only depending on its sign. Therefore, we compress the translated intensity difference vector $\mathbf{d}_\gamma$ instead of the weight vector $\mathbf{w}$. To this end, we treat $\mathbf{d}_\gamma$ as a graph signal on $\mathcal{G}_\ell$ and consider

$$\hat{\mathbf{d}}_\gamma = \mathbf{U}_\ell^T \mathbf{d}_\gamma.$$

As above, we can apply filtering methods on $\hat{\mathbf{d}}_\gamma$ in the graph spectral domain to encode $\mathbf{d}_\gamma$. The advantage of encoding $\mathbf{d}_\gamma$ is the high absolute values of similar and dissimilar values. If $\mathbf{d}_\gamma(e) \approx 0$ for an edge $e \in \mathcal{E}$, then the intensity difference along e is moderately large and the classification is not as important. If on the other hand $|\mathbf{d}_\gamma(e)| \gg 0$, then the intensity difference along e is either small or very large. It is crucial that those edges are assigned the correct weight,

i.e. $w_e = 1$ iff $\mathbf{d}_\gamma(e) \ll 0$ and $w_e = w_\mathrm{o}$ iff $\mathbf{d}_\gamma(e) \gg 0$.

In practice, we select only the absolute largest Fourier coefficients of $\hat{\mathbf{d}}_\gamma$ for encoding, resulting in a thresholded vector $\hat{\mathbf{d}}_{\gamma,\mathrm{th}}$. Due to the unchanged energy of a signal after a GFT in equation (4.11), we have

$$\left\|\hat{\mathbf{d}}_\gamma - \hat{\mathbf{d}}_{\gamma,\mathrm{th}}\right\|_2^2 = \left\|\mathbf{U}_\ell^T(\mathbf{d}_\gamma - \mathbf{d}_{\gamma,\mathrm{th}})\right\|_2^2 = \|\mathbf{d}_\gamma - \mathbf{d}_{\gamma,\mathrm{th}}\|_2^2 .$$

While this does not guarantee a perfect reconstruction, large absolute values of $\mathbf{d}_\gamma$ are more probable to be approximated good enough that the sign is correct.

5.8 Edge Weights

Having applied the discrete edge weight model in both of the previous sections, we now turn to computing w_o. The question is how to choose the weight $w_\mathrm{o} \in (0, 1)$ between weakly correlated pixels, so that the resulting GFT has the best possible performance. We derive the optimal weight w_o with methods from statistical analysis, where we show that the resulting GFT approximates the KLT for a class of significant signals.

The approach here is based on [45], where the optimal edge weight was derived for piecewise smooth images, where each block is assumed to only have one weak transition. We extend on that concept by considering more complex signals with several weak transitions.

Chapter 2.2 has shown the optimal properties of the KLT regarding decorrelation and compaction of energy contained in a signal $\mathbf{f} \in \mathbb{R}^N$. This is achieved by transforming $\mathbf{f}$ with the eigenvector matrix of the signal's covariance matrix $\mathbf{\Sigma_f}$. As discussed above, it is not feasible to apply the KLT directly. Consequently, we aim to construct a GFT that approximates the KLT. This is done by selecting a weight w_o so that the eigenvectors of the resulting Laplacian matrix $\mathbf{L}$ approximate the eigenvectors of $\mathbf{\Sigma_f}$.

As per our discussion in Section 5.4, a signal only contains strongly or weakly correlated neighbouring pixels. For simplicity, we will derive the optimal edge weight in one dimension and carry the weight over to two dimensions.

We construct a one-dimensional *model texture signal* $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$. If a successive pixel pair (x_{k-1}, x_k) is strongly correlated, we say they lie in a *smooth transition region* $\mathcal{S}$, i.e. $(k-1, k] \in \mathcal{S}$. For a weak correlation between x_{k-1} and x_k they lie in a *weak transition region* $\mathcal{T}$, i.e. $(k-1, k] \in \mathcal{T}$. This results in a partition of $(1, N]$,

$$(1, N] = \mathcal{S} \cup \mathcal{T} = \mathcal{S} \cup \left(\bigcup_{i=1}^{m} (k_i - 1, k_i] \right),$$

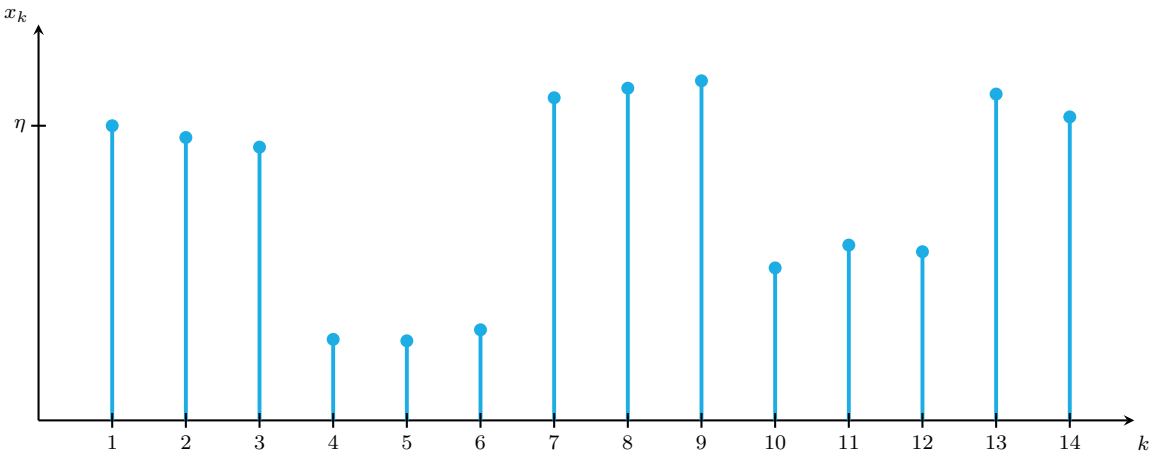
with m weak transitions at pixels $\{k_1, \ldots, k_m\} \subset \{2, \ldots, N\}$. We now model $\mathbf{x}$ by a first order autoregressive process.

Definition 5.7 *A* model texture signal $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$ *with* m *weak transitions at pixels* $\{k_1, \ldots, k_m\}$ *is defined by*

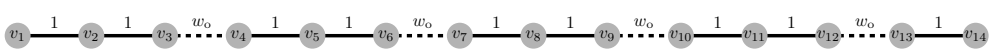
$$x_k = \begin{cases} \rho & \text{if } k = 1, \\ x_{k-1} + \varepsilon_k & \text{if } 1 < k \leq N \text{ and } (k-1, k] \in \mathcal{S}, \\ x_{k-1} + (-1)^j g_k + \varepsilon_k & \text{if } 1 < k \leq N \text{ and } (k-1, k] \in \mathcal{T}, \text{ where } k = k_j, \end{cases} \tag{5.14}$$

where $\varepsilon_k \sim \mathcal{N}(0, \sigma_s^2)$, $\rho \sim \mathcal{N}(0, \sigma_1^2)$ *and* $g_k \sim \mathcal{N}(\mu_g, \sigma_g^2)$ *for some* $\sigma_1^2, \sigma_s^2, \sigma_g^2 \in \mathbb{R}^+$ *and* $\mu_g \in \mathbb{R}^+$.

The first entry ρ in the signal corresponds to a corner vertex of a triangle, as it anchors the signal. As the signal to be compressed is given by $\mathbf{I}_t = \mathbf{I} - \widetilde{\mathbf{I}}$, the values of $\mathbf{I}_t$ are expected to be around zero, so we assume ρ to have mean $\mu_\rho = 0$. But since $\widetilde{\mathbf{I}}$ is constructed by the best approximation as in (3.3), the variance σ_1^2 is not zero. We are considering natural images, so even in smooth regions there are going to be small variations between neighbouring pixels. This is modeled by adding *independant and identically distributed* (i.i.d.) variations ε_k. Across the weak transition region $\mathcal{T}$ the variation is larger, modeled by adding i.i.d. random gaps $g_k \sim \mathcal{N}(\mu_g, \sigma_g^2)$. An example of a model texture signal $\mathbf{x}$ can be seen in Figure 5.17(a).



(a) Model texture signal $\mathbf{x} \in \mathbb{R}^{14}$.



(b) Chain graph $\mathcal{G}_\mathbf{x}$ corresponding to $\mathbf{x}$.

Figure 5.17: A texture signal $\mathbf{x} \in \mathbb{R}^{14}$ as defined in Definition 5.7 with four weak transitions at $k_1 = 4$, $k_2 = 7$, $k_3 = 10$ and $k_4 = 13$ along with its corresponding 2-connectivity chain graph $\mathcal{G}_\mathbf{x}$ with weight $w_o \in (0, 1)$.

In order to approximate the KLT with a GFT, we first compute the covariance matrix $\mathbf{\Sigma}_\mathbf{x}$ for a given model texture signal. We begin by stating a small technical Lemma.

Lemma 5.1 *The matrix* $\mathbf{F} \in \mathbb{R}^{N\times N}$*, given by*

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & \ddots & & \vdots \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}, \tag{5.15}$$

is invertable and $\mathbf{F}^{-1} \in \mathbb{R}^{N\times N}$ *is given by*

$$\mathbf{F}^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Proof: Since $\mathbf{F} \in \mathbb{R}^{N\times N}$ is a lower triangular matrix, we have $\det(\mathbf{F}) = 1$, thus $\mathbf{F}$ is invertible. Next, we consider left-multiplying with $\mathbf{F}^{-1}$ for a given matrix $\mathbf{A} = [a_{i,j}]_{1\leq i,j\leq N} \in \mathbb{R}^{N\times N}$. Since $\mathbf{F}^{-1}$ is a lower triangular matrix with constant value 1 on and below the diagonal, left-multiplying with it sums up all entries to the top of a given entry, i.e.

$$\left[\mathbf{F}^{-1}\mathbf{A}\right]_{i,j} = \sum_{k=1}^{i} a_{k,j}. \tag{5.16}$$

Letting $\mathbf{A} = \mathbf{F}$, we obtain the desired result

$$\left[\mathbf{F}^{-1}\mathbf{F}\right]_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

for all $1 \leq i, j \leq N$. □

Next, we compute the covariance matrix $\mathbf{\Sigma}_\mathbf{x}$. In the following, we assume that $\mathbf{x} \in \mathbb{R}^N$ is a model texture signal as defined in Definition 5.7 with m weak transitions at the indices $\{k_1, \ldots, k_m\} \subset \{3, \ldots, N-1\}$. Note that we require the first and last transition to be smooth for technical purposes.

Lemma 5.2 *The covariance matrix* $\mathbf{\Sigma}_\mathbf{x}$ *of a model texture signal* $\mathbf{x}$ *is given by*

$$[\mathbf{\Sigma}_\mathbf{x}]_{i,j} = \sigma_1^2 + (\min(i,j) - 1) \cdot \sigma_s^2 + \min(\mathrm{ind}_i, \mathrm{ind}_j) \cdot \sigma_g^2$$

for $1 \leq i, j \leq N$*, where* ind_i *denotes the nearest lower weak transition index, i.e.* $\mathrm{ind}_i = l$ *if* $k_l \leq i < k_{l+1}$*, with* $\mathrm{ind}_i = 0$ *if* $i < k_1$*.*

Proof: Let $\mathbf{x} \in \mathbb{R}^N$ be a model texture signal as above. We expand $\mathbf{x}$ by

$$\begin{aligned} x_1 &= \rho, \\ x_k - x_{k-1} &= \begin{cases} \varepsilon_k + (-1)^j g_k & \text{if } k = k_j, \\ \varepsilon_k & \text{otherwise} \end{cases} \end{aligned}$$

for $1 < k \leq N$. This can be expressed in matrix form via

$$\mathbf{F}\mathbf{x} = \mathbf{b}, \tag{5.17}$$

where $\mathbf{F} \in \mathbb{R}^{N \times N}$ is defined as in (5.15) and $\mathbf{b} \in \mathbb{R}^N$ is given by

$$\mathbf{b} = \begin{pmatrix} \rho \\ \varepsilon_2 \\ \vdots \\ g_{k_1} + \varepsilon_{k_1} \\ \vdots \\ -g_{k_2} + \varepsilon_{k_2} \\ \vdots \\ g_{k_3} + \varepsilon_{k_3} \\ \vdots \\ \varepsilon_N \end{pmatrix}, \tag{5.18}$$

with the variations ε_k in smooth regions. Lemma 5.1 in combination with (5.17) lets us express $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{F}^{-1}\mathbf{b}.$$

The covariance matrix $\mathbf{\Sigma_x}$ now computes as

$$\begin{aligned} \mathbf{\Sigma_x} &= \mathrm{E}\left[(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^T\right] \\ &= \mathrm{E}\left[\mathbf{x}\mathbf{x}^T\right] - \boldsymbol{\mu}_\mathbf{x}\boldsymbol{\mu}_\mathbf{x}^T, \end{aligned} \tag{5.19}$$

where $\boldsymbol{\mu}_\mathbf{x} = (\mu_1, \ldots, \mu_N)^T = \mathrm{E}[\mathbf{x}]$. Note that we applied the linearity property of expectations in the second equation, see [68].

Since $\mathrm{E}[\rho] = \mathrm{E}[\varepsilon_l] = 0$ for $2 \leq l \leq N$ and $\mathrm{E}\left[g_{k_j}\right] = \mu_g$ for $k_j \in \{k_1, \ldots, k_m\}$, we have

$$\boldsymbol{\mu}(k) = \begin{cases} -\mu_g & k \in \bigcup_{l=1}^{\lfloor \mathrm{m}/2 \rfloor} \{k_{2l-1}, \ldots k_{2l} - 1\} \\ 0 & \text{otherwise.} \end{cases}$$

for all $1 \leq k \leq N$. Therefore, $\boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T \in \mathbb{R}^{N\times N}$ is given by

$$\left[\boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T\right]_{i,j} = \begin{cases} \mu_g^2 & \text{if } i,j \in \bigcup_{l=1}^{\lfloor m/2 \rfloor} \{k_{2l-1}, \dots k_{2l}-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{5.20}$$

The equation (5.19) in combination with (5.17) implies

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{x}} &= \mathrm{E}\left[\mathbf{F}^{-1}\mathbf{b}\mathbf{b}^T(\mathbf{F}^{-1})^T\right] - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T \\ &= \mathbf{F}^{-1}\mathrm{E}\left[\mathbf{b}\mathbf{b}^T\right](\mathbf{F}^{-1})^T - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T. \end{aligned} \tag{5.21}$$

To compute $\mathrm{E}\left[\mathbf{b}\mathbf{b}^T\right]$, we first note that all ε_k, $1 < k \leq N$ and g_{k_l}, $1 \leq l \leq m$ are pairwise independent. Hence, we have

$$\begin{aligned} \mathrm{E}\left[\varepsilon_i\varepsilon_j\right] &= \delta_{ij}\sigma_s^2 \quad \text{for } 1 < i \leq j \leq N, \\ \mathrm{E}\left[(\varepsilon_{k_i} \pm g_{k_i})(\varepsilon_{k_j} \pm g_{k_j})\right] &= \begin{cases} \sigma_s^2 + \mu_g^2 + \sigma_g^2 & \text{if } 1 \leq i = j \leq m, \\ \pm\mu_g^2 & \text{if } 1 \leq i \neq j \leq m, \end{cases} \end{aligned} \tag{5.22}$$

where we have again applied the linearity property of expectations in the second equation. Inserting $\mathbf{b}$ as in (5.18) and applying (5.22) yields

$$\begin{aligned} \left[\mathrm{E}\left[\mathbf{b}\mathbf{b}^T\right]\right]_{i,j} &= \begin{cases} \mathrm{E}\left[\rho^2\right] & \text{if } i = j = 1, \\ \mathrm{E}\left[\varepsilon_i\varepsilon_j\right] & \text{if } i,j \notin \{k_1,\dots,k_m\}, \\ \mathrm{E}\left[\left(\varepsilon_i + (-1)^{l_i}\mu_g\right)\left(\varepsilon_j + (-1)^{l_j}\mu_g\right)\right] & \text{if } i = k_{l_i}, j = k_{l_j} \in \{k_1,\dots,k_m\}, \\ \mathrm{E}\left[\left(\varepsilon_i + (-1)^{l_i}\mu_g\right)\varepsilon_j\right] & \text{if } i = k_{l_i} \in \{k_1,\dots,k_m\} \not\ni j, \\ \mathrm{E}\left[\varepsilon_i\left(\varepsilon_j + (-1)^{l_j}\mu_g\right)\right] & \text{if } j = k_{l_j} \in \{k_1,\dots,k_m\} \not\ni i, \end{cases} \\ &= \begin{cases} \sigma_1^2 & \text{if } i = j = 1, \\ \delta_{ij}\sigma_s^2 & \text{if } i,j \notin \{k_1,\dots,k_m\}, \\ \sigma_s^2 + \mu_g^2 + \sigma_g^2 & \text{if } i = j \in \{k_1,\dots,k_m\}, \\ (-1)^{l_i+l_j}\mu_g^2 & \text{if } i = k_{l_i} \neq j = k_{l_j} \in \{k_1,\dots,k_m\}, \\ 0 & \text{otherwise,} \end{cases} \end{aligned}$$

where δ_{ij} is the *Kronecker delta* satisfying

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Written in matrix form, we have

$$
\begin{aligned}
E\left[\mathbf{b}\mathbf{b}^T\right] &= \begin{pmatrix}
\sigma_1^2 & & & & & & & & \\
& \sigma_s^2 & & & & & & & \\
& & \ddots & & & & & & \\
& & & \sigma_s^2+\mu_g^2+\sigma_g^2 & & -\mu_g^2 & & \mu_g^2 & \\
& & & & \ddots & & & & \\
& & & -\mu_g^2 & & \sigma_s^2+\mu_g^2+\sigma_g^2 & & -\mu_g^2 & \\
& & & & & & \ddots & & \\
& & & \mu_g^2 & & -\mu_g^2 & & \sigma_s^2+\mu_g^2+\sigma_g^2 & \\
& & & & & & & & \ddots \\
& & & & & & & & & \sigma_s^2
\end{pmatrix} \\
&= \begin{pmatrix}
\sigma_1^2 & & & & & & \\
& \sigma_s^2 & & & & & \\
& & \ddots & & & & \\
& & & \sigma_s^2+\sigma_g^2 & & & \\
& & & & \ddots & & \\
& & & & & \sigma_s^2+\sigma_g^2 & \\
& & & & & & \ddots
\end{pmatrix} + \begin{pmatrix}
0 & & & & & & \\
& \ddots & & & & & \\
& & \mu_g^2 & & -\mu_g^2 & & \mu_g^2 \\
& & & \ddots & & & \\
& & -\mu_g^2 & & \mu_g^2 & & -\mu_g^2 \\
& & & & & \ddots & \\
& & \mu_g^2 & & -\mu_g^2 & & \mu_g^2 \\
& & & & & & \ddots
\end{pmatrix} \\
&=: \mathbf{S} + \mathbf{M},
\end{aligned}
\tag{5.23}
$$

where the dots represent σ_s^2 in $\mathbf{S}$ and 0 in $\mathbf{M}$. In order to compute $\mathbf{\Sigma_x}$, we need to evaluate (5.21). To this end, we split $\mathrm{E}\left[\mathbf{b}\mathbf{b}^T\right] = \mathbf{S} + \mathbf{M}$ into two parts as in (5.23). $\mathbf{M} \in \mathbb{R}^{N\times N}$ comprises the expected values μ_g and $\mathbf{S} \in \mathbb{R}^{N\times N}$ the rest, so that

$$
[\mathbf{M}]_{i,j} = \begin{cases}
(-1)^{l_i+l_j}\mu_g^2 & \text{if } i = k_{l_i}, j = k_{l_j} \in \{k_1, \ldots, k_m\}, \\
0 & \text{otherwise,}
\end{cases}
$$

$$
[\mathbf{S}]_{i,j} = \begin{cases}
\sigma_1^2 & \text{if } i = j = 1, \\
\sigma_s^2 + \sigma_g^2 & \text{if } i = j \in \{k_1, \ldots, k_m\}, \\
\sigma_s^2 & \text{if } 1 < i = j \notin \{k_1, \ldots, k_m\}, \\
0 & \text{otherwise.}
\end{cases}
$$

To compute $\mathbf{F}^{-1}\mathbf{S}\left(\mathbf{F}^{-1}\right)^T$ and $\mathbf{F}^{-1}\mathbf{M}\left(\mathbf{F}^{-1}\right)^T$, we again consider left-multiplying with $\mathbf{F}^{-1}$ and right-multiplying with $\left(\mathbf{F}^{-1}\right)^T$ for a given matrix $\mathbf{A} = [a_{i,j}]_{1\le i,j\le N} \in \mathbb{R}^{N\times N}$. As in the proof of Lemma 5.1, we have

$$
\left[\mathbf{F}^{-1}\mathbf{A}\right]_{i,j} = \sum_{k=1}^{i} a_{k,j}. \tag{5.24}
$$

Analogously, since $\left(\mathbf{F}^{-1}\right)^T$ is an upper triangular matrix with constant value one on and above the diagonal, right-multiplying with it sums up all entries to the left of a given entry, i.e.

$$
\left[\mathbf{A}\left(\mathbf{F}^{-1}\right)^T\right]_{i,j} = \sum_{l=1}^{j} a_{i,l}. \tag{5.25}
$$

Combining (5.16) and (5.25) shows that $\mathbf{F}^{-1}\mathbf{A}\left(\mathbf{F}^{-1}\right)^T$ sums up all entries of $\mathbf{A}$ to the top and

left of a given entry:

$$\left[\mathbf{F}^{-1}\mathbf{A}\left(\mathbf{F}^{-1}\right)^T\right]_{i,j} = \sum_{k=1}^{i}\sum_{l=1}^{j} a_{k,l}.$$

Applying this multiplication to $\mathbf{S}$ and $\mathbf{M}$ yields

$$\begin{aligned}\left[\mathbf{F}^{-1}\mathbf{S}\left(\mathbf{F}^{-1}\right)^T\right]_{i,j} &= \sigma_1^2 + (\min(i,j)-1)\sigma_s^2 + \min(\mathrm{ind}_i,\mathrm{ind}_j)\sigma_g^2,\\ \left[\mathbf{F}^{-1}\mathbf{M}\left(\mathbf{F}^{-1}\right)^T\right]_{i,j} &= \begin{cases}\mu_g^2 & \text{if } i,j \in \bigcup_{l=1}^{\lfloor m/2\rfloor}\{k_{2l-1},\dots k_{2l}-1\}\\ 0 & \text{otherwise,}\end{cases}\end{aligned} \tag{5.26}$$

where $\mathrm{ind}_i = l$ if $k_l \le i < k_{l+1}$, with $\mathrm{ind}_i = 0$ if $i < k_1$. From (5.20) and (5.26) we see that $\boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T = \mathbf{F}^{-1}\mathbf{M}\left(\mathbf{F}^{-1}\right)^T$. This, in combination with (5.21) implies the desired result

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{x}} &= \mathbf{F}^{-1}\mathrm{E}\left[\mathbf{b}\mathbf{b}^T\right]\left(\mathbf{F}^{-1}\right)^T - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T\\ &= \mathbf{F}^{-1}\mathbf{S}\left(\mathbf{F}^{-1}\right)^T + \mathbf{F}^{-1}\mathbf{M}\left(\mathbf{F}^{-1}\right)^T - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^T\\ &= \mathbf{F}^{-1}\mathbf{S}\left(\mathbf{F}^{-1}\right)^T\\ &= \begin{pmatrix}
\sigma_1^2 & \sigma_1^2 & \sigma_1^2 & \cdots & \sigma_1^2 & \cdots & \sigma_1^2 & \cdots & \sigma_1^2\\
\sigma_1^2 & \sigma_1^2+\sigma_s^2 & \sigma_1^2+\sigma_s^2 & \cdots & \sigma_1^2+\sigma_s^2 & \cdots & \sigma_1^2+\sigma_s^2 & \cdots & \sigma_1^2+\sigma_s^2\\
\sigma_1^2 & \sigma_1^2+\sigma_s^2 & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+2\sigma_s^2\\
\vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & & \vdots\\
\sigma_1^2 & \sigma_1^2+\sigma_s^2 & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+(k_1-1)\sigma_s^2+\sigma_g^2 & \cdots & \sigma_1^2+(k_1-1)\sigma_s^2+\sigma_g^2 & \cdots & \sigma_1^2+(k_1-1)\sigma_s^2+\sigma_g^2\\
\vdots & \vdots & \vdots & & \vdots & \ddots & \vdots & & \vdots\\
\sigma_1^2 & \sigma_1^2+\sigma_s^2 & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+(k_1-1)\sigma_s^2+\sigma_g^2 & \cdots & \sigma_1^2+(k_2-1)\sigma_s^2+2\sigma_g^2 & \cdots & \sigma_1^2+(k_2-1)\sigma_s^2+2\sigma_g^2\\
\vdots & \vdots & \vdots & & \vdots & & \vdots & \ddots & \vdots\\
\sigma_1^2 & \sigma_1^2+\sigma_s^2 & \sigma_1^2+2\sigma_s^2 & \cdots & \sigma_1^2+(k_1-1)\sigma_s^2+\sigma_g^2 & \cdots & \sigma_1^2+(k_2-1)\sigma_s^2+2\sigma_g^2 & \cdots & \sigma_1^2+(n-1)\sigma_s^2+m\sigma_g^2
\end{pmatrix}.\end{aligned}$$

□

In order to relate the KLT to the GFT, we compute the corresponding *precision matrix* $\mathbf{P}_{\mathbf{x}}$, i.e. the inverse of $\boldsymbol{\Sigma}_{\mathbf{x}}$.

Lemma 5.3 *For a signal $\mathbf{x} \in \mathbb{R}^N$ as defined in* (5.14)*, the precision matrix $\mathbf{P}_{\mathbf{x}} = (\boldsymbol{\Sigma}_{\mathbf{x}})^{-1}$ is a tridiagonal matrix given by*

$$[\mathbf{P}_{\mathbf{x}}]_{i,j} = \begin{cases}
0 & \text{if } |i-j|>1,\\
\frac{1}{\sigma_1^2}+\frac{1}{\sigma_s^2} & \text{if } i=j=1,\\
\frac{1}{\sigma_s^2} & \text{if } i=j=N,\\
2\frac{1}{\sigma_s^2} & \text{if } 1<i=j<N, i\notin\{k_1,\dots,k_m\},\\
\frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2} & \text{if } i=j\in\{k_1-1,k_1,\dots,k_m-1,k_m\},\\
-\frac{1}{\sigma_g^2+\sigma_s^2} & \text{if } i+1=j\in\{k_1,\dots,k_m\} \text{ or } j+1=i\in\{k_1,\dots,k_m\},\\
-\frac{1}{\sigma_s^2} & \text{otherwise.}
\end{cases}$$

In matrix form we have

$$\mathbf{P_x} = \begin{pmatrix} \frac{1}{\sigma_1^2}+\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & & & & & & & & & \\ -\frac{1}{\sigma_s^2} & 2\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & & \\ & & -\frac{1}{\sigma_s^2} & 2\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & & & & & & \\ & & & -\frac{1}{\sigma_s^2} & \frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2} & -\frac{1}{\sigma_g^2+\sigma_s^2} & & & & & \\ & & & & -\frac{1}{\sigma_g^2+\sigma_s^2} & \frac{1}{\sigma_g^2+\sigma_s^2}+\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & & & & \\ & & & & & -\frac{1}{\sigma_s^2} & 2\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & & & \\ & & & & & & \ddots & \ddots & \ddots & & \\ & & & & & & & -\frac{1}{\sigma_s^2} & 2\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & \\ & & & & & & & & -\frac{1}{\sigma_s^2} & \frac{1}{\sigma_s^2} & \end{pmatrix},$$

with jumps between rows $k_j - 1$ and k_j for all $k_j \in \{k_1, \ldots, k_m\}$.

Proof: Let $\mathbf{x} \in \mathbb{R}^N$ be a model texture signal as above. We first show $\mathbf{P_x}\mathbf{\Sigma_x} = \mathbf{Id}_N$, where $\mathbf{Id}_N$ is the $N \times N$ identity matrix. To this end, we compute $[\mathbf{P_x}\mathbf{\Sigma_x}]_{i,j}$ for all $1 \leq i, j \leq N$. Due to the specific structure of $\mathbf{P_x}$, we distinguish different cases for $1 \leq i \leq N$. In the following, we denote the i$^{\text{th}}$ row of $\mathbf{P_x}$ by $\text{row}_i(\mathbf{P_x})$ and the j$^{\text{th}}$ column analogously by $\text{col}_j\mathbf{P_x}$.

Case 1: $i = 1$. In this case we have

$$\text{row}_i\left(\mathbf{P_x}\right) = \begin{pmatrix} \frac{1}{\sigma_s^2}+\frac{1}{\sigma_1^2} & -\frac{1}{\sigma_s^2} & 0 & \cdots & 0 \end{pmatrix}.$$

Therefore, multiplication with $\text{col}_j\left(\mathbf{\Sigma_x}\right)$ for $1 \leq j \leq N$ yields

$$[\mathbf{P_x}\mathbf{\Sigma_x}]_{1,j} = \begin{cases} \left(\frac{1}{\sigma_s^2}+\frac{1}{\sigma_1^2}\right)\sigma_1^2 - \frac{1}{\sigma_s^2}\sigma_1^2 = 1 & \text{if } j = 1, \\ \left(\frac{1}{\sigma_s^2}+\frac{1}{\sigma_1^2}\right)\sigma_1^2 - \frac{1}{\sigma_s^2}\left(\sigma_1^2+\sigma_s^2\right) = 0 & \text{for } 2 \leq j \leq N. \end{cases}$$

Case 2: $i = N$. In this case we have

$$\text{row}_i\left(\mathbf{P_x}\right) = \begin{pmatrix} 0 & \cdots & 0 & -\frac{1}{\sigma_s^2} & \frac{1}{\sigma_s^2} \end{pmatrix}.$$

Therefore, multiplication with $\text{col}_j\left(\mathbf{\Sigma_x}\right)$ for $1 \leq j \leq N$ yields

$$[\mathbf{P_x}\mathbf{\Sigma_x}]_{N,j} = \begin{cases} -\frac{1}{\sigma_s^2}\left(\sigma_1^2+(n-2)\sigma_s^2+m\sigma_g^2\right)+\frac{1}{\sigma_s^2}\left(\sigma_1^2+\sigma_s^2(n-1)+m\sigma_g^2\right) = 1 & \text{if } j = N \\ -\frac{1}{\sigma_s^2}^2\left(\sigma_1^2+(j-1)\sigma_s^2+m\sigma_g^2\right)+\frac{1}{\sigma_s^2}\left(\sigma_1^2+\sigma_s^2(j-1)+m\sigma_g^2\right) = 0 & \text{for } 1 \leq j \leq N-1. \end{cases}$$

Case 3: $i = k_l - 1$ for $k_l \in \{k_1, \ldots, k_m\}$. In this case we have

$$\text{row}_i\left(\mathbf{P_x}\right) = \begin{pmatrix} 0 & \cdots & 0 & -\frac{1}{\sigma_s^2} & \frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+1} & -\frac{1}{\sigma_g^2+1} & 0 & \ldots & 0 \end{pmatrix}.$$

Therefore, multiplication with $\mathrm{col}_i(\mathbf{\Sigma_x})$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,i} = -\frac{1}{\sigma_s^2}\left(\sigma_1^2+(k_l-3)\sigma_s^2\right)+\left(\frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2}\right)\left(\sigma_1^2+(k_l-2)\sigma_s^2\right) + \frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+(k_l-2)\sigma_s^2\right)=1.$$

Multiplication with $\mathrm{col}_j(\mathbf{\Sigma_x})$ for $1 \leq j < i$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,j} = -\frac{1}{\sigma_s^2}\left(\sigma_1^2+(j-1)\sigma_s^2\right)+\left(\frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2}\right)\left(\sigma_1^2+(j-1)\sigma_s^2\right) - \frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+(j-1)\sigma_s^2\right)=0,$$

and multiplication with $\mathrm{col}_j(\mathbf{\Sigma_x})$ for $i < j \leq N$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,j} = -\frac{1}{\sigma_s^2}\left(\sigma_1^2+(i-2)\sigma_s^2\right)+\left(\frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2}\right)\left(\sigma_1^2+(i-1)\sigma_s^2\right) - \frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+i\sigma_s^2+\sigma_g^2\right)=1+\frac{1}{\sigma_g^2+\sigma_s^2}\left(-\sigma_g-\sigma_s^2\right)=0.$$

Case 4: $i = k_l$ for $k_l \in \{k_1, \ldots, k_m\}$. In this case we have

$$\mathrm{row}_i(\mathbf{P_x}) = \begin{pmatrix} 0 & \cdots & 0 & -\frac{1}{\sigma_g^2+\sigma_s^2} & \frac{1}{\sigma_s^2}+\frac{1}{\sigma_g^2+\sigma_s^2} & -\frac{1}{\sigma_s^2} & 0 & \ldots & 0 \end{pmatrix}.$$

Therefore, multiplication with $\mathrm{col}_i(\mathbf{\Sigma_x})$ yields

$$\begin{aligned}[\mathbf{P_x\Sigma_x}]_{i,i} =& -\frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+(k_l-2)\sigma_s^2\right)+\left(\frac{1}{\sigma_g^2+\sigma_s^2}+\frac{1}{\sigma_s^2}\right)\left(\sigma_1^2+(k_l-1)\sigma_s^2+\sigma_g^2\right)\\ &-\frac{1}{\sigma_s^2}\left(\sigma_1^2+(k_l-1)\sigma_s^2+\sigma_g^2\right)\\ =&\frac{1}{\sigma_g^2+\sigma_s^2}\left(-\sigma_1^2-(k_l-2)\sigma_s^2+\sigma_1^2+(k_l-1)\sigma_s^2+\sigma_g^2\right)\\ =&\frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_s^2+\sigma_g^2\right)=1.\end{aligned}$$

Multiplication with $\mathrm{col}_j(\mathbf{\Sigma_x})$ for $1 \leq j < i$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,j} = -\frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+(j-1)\sigma_s^2\right)+\left(\frac{1}{\sigma_g^2+\sigma_s^2}+\frac{1}{\sigma_s^2}\right)\left(\sigma_1^2+(j-1)\sigma_s^2\right) - \frac{1}{\sigma_s^2}\left(\sigma_1^2+(j-1)\sigma_s^2\right)=0,$$

and multiplication with $\mathrm{col}_j(\mathbf{\Sigma_x})$ for $i < j \leq N$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,j} = -\frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_1^2+(i-2)\sigma_s^2\right)+\left(\frac{1}{\sigma_g^2+\sigma_s^2}+\frac{1}{\sigma_s^2}\right)\left(\sigma_1^2+(i-1)\sigma_s^2+\sigma_g^2\right) - \frac{1}{\sigma_s^2}\left(\sigma_1^2+i\sigma_s^2+\sigma_g^2\right)=\frac{1}{\sigma_g^2+\sigma_s^2}\left(\sigma_s^2+\sigma_g^2\right)-1=0.$$

Case 5: For all other choices of i we have

$$\mathrm{row}_i\left(\mathbf{P_x}\right) = \begin{pmatrix} 0 & \cdots & 0 & -\frac{1}{\sigma_s^2} & 2\frac{1}{\sigma_s^2} & -\frac{1}{\sigma_s^2} & 0 & \dots & 0 \end{pmatrix}.$$

Therefore, multiplication with $\mathrm{col}_j\left(\mathbf{\Sigma_x}\right)$ for $1 \leq j \leq N$ yields

$$[\mathbf{P_x\Sigma_x}]_{i,j} = \begin{cases} -\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (i-2)\sigma_s^2\right) + 2\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (i-1)\sigma_s^2\right) - \frac{1}{\sigma_s^2}\left(\sigma_1^2 + (i-1)\sigma_s^2\right) = 1 & \text{if } i = j, \\ -\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (i-2)\sigma_s^2\right) + 2\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (i-1)\sigma_s^2\right) - \frac{1}{\sigma_s^2}\left(\sigma_1^2 + i\sigma_s^2\right) = 0 & \text{for } i < j, \\ -\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (j-1)\sigma_s^2\right) + 2\frac{1}{\sigma_s^2}\left(\sigma_1^2 + (j-1)\sigma_s^2\right) - \frac{1}{\sigma_s^2}\left(\sigma_1^2 + (j-1)\sigma_s^2\right) = 0 & \text{for } i > j. \end{cases}$$

Combining these five cases, we obtain

$$[\mathbf{P_x\Sigma_x}]_{i,j} = \delta_{ij},$$

or $\mathbf{P_x\Sigma_x} = \mathbf{Id}_N$ in matrix form. Since $\mathbf{\Sigma_x}$ and $\mathbf{P_x}$ are symmetric, we obtain the desired result:

$$\mathbf{P_x\Sigma_x} = \mathbf{Id}_N = \left(\mathbf{P_x\Sigma_x}\right)^T = \mathbf{\Sigma_x^T P_x^T} = \mathbf{\Sigma_x P_x}.$$

□

Finally, we construct the 2-connectivity chain graph $\mathcal{G}_\mathbf{x}$ corresponding to a model texture signal $\mathbf{x} \in \mathbb{R}^N$. It consists of N vertices $\mathcal{V} = \{v_1, \dots, v_N\}$, where each vertex is connected to its direct neighbours, i.e. $\mathcal{E} = \{(v_i, v_{i+1}) : 1 \leq i < N\}$. Edges across a weak transition region are assigned weight $\frac{1}{\sigma_g^2+\sigma_s^2}$, i.e. $W_{k_j-1,k_j} = \frac{1}{\sigma_g^2+\sigma_s^2}$ if $k_j \in \{k_1, \dots k_m\}$. All other edges are assigned weight $\frac{1}{\sigma_s^2}$, i.e. $W_{j-1,j} = \frac{1}{\sigma_s^2}$ for $2 \leq j \leq N$ with $j \notin \{k_1, \dots, k_m\}$). An example for $\mathcal{G}_\mathbf{x}$ can be seen in figure 5.17(b).

Comparing the graph Laplacian matrix $\mathbf{L_x}$ derived from $\mathcal{G}_\mathbf{x}$ with the precision matrix $\mathbf{P_x}$ in Lemma 5.3, we observe that they are equivalent except for the first entry, where

$$[\mathbf{P_x}]_{1,1} = [\mathbf{L_x}]_{1,1} + \frac{1}{\sigma_1^2}. \tag{5.27}$$

But since the variance σ_1^2 of the first entry is fairly larger than zero, $\mathbf{P_x}$ and $\mathbf{L_x}$ are approximately equivalent. $\mathbf{P_x}$ and $\mathbf{\Sigma_x}$ share the same set of eigenvectors, see [94], which form the basis of the KLT. Consequently, the derived GFT approximates the KLT for model texture signals $\mathbf{x}$ as defined above.

We finally norm $\mathcal{G}_\mathbf{x}$ by multiplying its weight matrix $\mathbf{W}$ by σ_s^2, so that $W_{j-1,j} = 1$ for $2 \leq j \leq N$ with $j \notin \{k_1, \dots, k_m\}$. Hence, in order to approximate the optimal decorrelation property of the KLT, we set the weight for weak correlation as

$$w_\mathrm{o} = \frac{\sigma_s^2}{\sigma_g^2 + \sigma_s^2}. \tag{5.28}$$

We show how to choose $w_\mathrm{o} \in (0, 1)$ for applicable blocks in Section 6.1.2.

Chapter 6

Experimental Results

In the previous chapter we have introduced the theoretical foundations of the ATGSP post-processing scheme, which improves the visual quality of specific triangular image blocks in an adaptive thinning reconstruction. It classifies them as either smooth or textured, and constructs adaptive weighted graphs on the latter. They are designed so that the texture image signal is smooth in regard to them. Transforming the signals with a respective GFT results in a sparse Fourier spectrum, with most information concentrated in few Fourier coefficients.

Building on this theoretical basis, we show how to implement the ATGSP scheme in this chapter. We begin by describing the experimental setup, starting with the selection of significant triangles that constitute $\mathcal{T}_{\text{sig}}$. For textured triangular image blocks, we estimate the optimal edge weight and construct an adaptive matrix. Afterwards, the coefficients are thresholded and quantized. Having described the setup, we show examples of natural images compressed by ATGSP.

We remark that the ATGSP scheme is still in a very early developement stage, therefore no complete implementation is given. In particular, we did not interfere in the selection process of significant pixels in the AT algorithm, which yields suboptimal triangulations for our scheme. Instead, we focus on the transformation process, which was implemented in Matlab. The ATGSP coefficients are computed and quantized, but not actually encoded. Therefore, we compare the ATGSP scheme exclusively with unaltered AT reconstructions.

6.1 Experimental Setup

Building on the theoretical basis introduced in Chapter 5, we show the practical implementation of our scheme to increase the visual quality of $\widetilde{\mathbf{I}}$. For the whole section we let $\mathbf{I}$ be a digital image on a pixel set P. The AT reconstruction $\widetilde{\mathbf{I}}$ is given over the Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$ of a set $\mathrm{P}_n \in \mathrm{P}$ comprising n significant pixels.

We begin by describing the selection process for the set of significant triangles $\mathcal{T}_{\text{sig}}$. Afterwards we discuss how the compression of $\mathbf{I}_t = \mathbf{I} - \widetilde{\mathbf{I}}$ for a single significant triangle $T \in \mathcal{T}_{\text{sig}} \subset \mathcal{D}(\mathrm{P}_n)$ is performed in practice.

As shown in Section 5.3, we construct a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$ on T and let $\mathbf{f}$ be the graph signal of $\mathbf{I}_t$ on $\mathcal{G}$. While the choice of vertices $\mathcal{V} = T$ is identical for all $T \in \mathcal{T}_{\text{sig}}$, more care has to be given to the construction of $\mathcal{E}$ and $\mathbf{W}$. Finally, we discuss how to threshold and quantize the resulting coefficients.

The quality of the compression in ATGSP is controlled via a level of quality $1 \leq q \leq 100$. Low values of q indicate lower quality along with higher compression, whereas high values indicate better quality at the cost of lower compression. It influences the compression quality twofold: it affects (i) the selection of significant triangles and (ii) the quantization on image blocks. Similarly to the JPEG compression scheme in Section 2.3, point (ii) is affected through a quality factor $q_F \in \mathbb{R}^+$ derived from q, given by

$$q_F = \begin{cases} \frac{60}{110-q} & \text{if } q > 50, \\ \frac{q+25}{75} & \text{if } q \leq 50. \end{cases}$$

This leads to a quality factor of $1/3 \leq q_F \leq 6$.

6.1.1 Selection of Significant Triangles

Recall from Section 5.2 that we aim to improve the visual quality of the AT reconstruction $\widetilde{\mathbf{I}}$ by adding information of $\mathbf{I}_t = \mathbf{I} - \widetilde{\mathbf{I}}$ over selected triangular image blocks. A crucial point for image improvement via graph spectral processing as introduced in Chapter 5 is therefore the selection of significant triangles. Since the AT reconstruction $\widetilde{\mathbf{I}}$ excels at preserving geometrical features and sharp edges, we focus on heavily textured regions in $\mathbf{I}$ that are not represented well in $\widetilde{\mathbf{I}}$. To this end, we select those triangles $T \in \mathcal{D}(\mathrm{P}_n)$ where $\mathbf{I}_t$ contains fine scale details.

Chapter 5.5 introduced the structure tensor S_T as a tool to identify the textural content of an image block T through its eigenvalues $\lambda_1 \geq \lambda_2 \geq 0$. Based on the magnitude of λ_1 and coherence c_T, the block is classified into one of three classes. Smooth blocks in Class 1 are characterized by a small eigencalue $\lambda_1 \approx 0$. Although λ_1 is large for both other classes, blocks with a principal dominant gradient in Class 2 additionally have a large coherence. Lastly, blocks in Class 3 have a small coherence and feature a complex structure that is not described by a single vector.

In order to classify each image block $T \in \mathcal{D}(\mathrm{P}_n)$, we introduce two thresholds $\varphi_\lambda, \varphi_c \in \mathbb{R}^+$. First we partition the set of all triangular image blocks $\mathcal{D}(\mathrm{P}_n) = \mathcal{T}_\mathrm{S} \cup \mathcal{T}_\mathrm{T}$ into a set of smooth $\mathcal{T}_\mathrm{S}$ and textured triangles $\mathcal{T}_\mathrm{T}$ based on λ_1. In particular, $T \in \mathcal{T}_\mathrm{S}$ iff $\lambda_1 \leq \varphi_\lambda$ and $T \in \mathcal{T}_\mathrm{T}$ iff $\lambda_1 > \varphi_\lambda$ for a specific image block $T \in \mathcal{D}(\mathrm{P}_n)$. For later sections it is necessary to further subdivide the set of textured image blocks $\mathcal{T}_\mathrm{T} = \mathcal{T}_\mathrm{D} \cup \mathcal{T}_\mathrm{C}$. All blocks T with a dominant principal gradient are assigned to $\mathcal{T}_\mathrm{D}$, that is $T \in \mathcal{T}_\mathrm{D}$ iff $c_T \geq \varphi_c$. On the other hand, blocks T with complex textures are assigned to $\mathcal{T}_\mathrm{C}$, that is $T \in \mathcal{T}_\mathrm{C}$ iff $c_T < \varphi_c$.

In order to set the threshold values for the implementation, recall that in Definition 5.5 the structure tensor of a block T was defined with a normalization factor $1/|T|$. Thus, φ_λ can be chosen to be independent of the size of each image block. In practice, a value of $\varphi_\lambda = 35$ gives a satisfactory distinction between smooth and textured image blocks.

When comparing the coding cost for adaptive graphs, triangles with complex textures are significantly more expensive than triangles in Class 2, where the graph structure depends simply on one of four quantized directions. Therefore, we set $\varphi_c = 0.1$ relatively low, so that most textured image blocks are assigned Class 2 with a cheaply encoded graph. Only if the textures are very complex and isotropic do we assign Class 3.

Having established the partition into smooth and textured triangles, we turn to the choice of $\mathcal{T}_{\text{sig}}$. As discussed above, we prioritize textured triangles, so most significant triangles will be elements of $\mathcal{T}_{\text{T}}$. Even though all image blocks in $\mathcal{T}_{\text{T}}$ are textured for our purposes, they have a different influence on the visual perception of the reconstruction. To this end, we first introduce a *local SSIM*.

As discussed in Section 5.2, we measure visual quality of a reconstruction via the SSIM introduced in Section 2.1. Since we consider small, localized image blocks, computing the SSIM based on the whole image would be too expensive. Therefore, we compute the estimated significance $\text{SSIM}_{\mathcal{L}}(T)$ of a textured image block $T \in \mathcal{T}_{\text{T}}$ locally. This measure is further designed to weigh the trade-off between two main criteria: distortion versus block size. Large distortions over a block should obviously have a high priority, but we need to balance this with the block size. An equal amount of coefficients on a large block hold more information than the same amount of coefficients on a smaller block. Hence, it may be worthwile to consider a slightly less textured but larger image block instead. The conventional SSIM value does not take signal size into account, and thus a single outlier pixel in a small block may have a disproportionate significance.

We achieve a good balance between both criteria by weighing the local SSIM of the original image $\mathbf{I}$ and the AT reconstruction $\widetilde{\mathbf{I}}$ on T by the number of vertices $N = |T|$

$$\text{SSIM}_{\mathcal{L}}(T) = N \cdot \left(1 - \text{SSIM}\left(\mathbf{I}_T, \widetilde{\mathbf{I}}_T\right)\right).$$

Large values of $\text{SSIM}_{\mathcal{L}}$ indicate a large block combined with a bad visual quality of the AT reconstruction. Hence, we prioritize blocks with larger values of $\text{SSIM}_{\mathcal{L}}$. To this end, we order the elements of $\mathcal{T}_{\text{T}}$ in descending order of their local SSIM value, so that $\mathcal{T}_{\text{T}} = \{T_k : 1 \leq k \leq |\mathcal{T}_{\text{T}}|\}$ with $\text{SSIM}_{\mathcal{L}}(T_i) \geq \text{SSIM}_{\mathcal{L}}(T_j)$ for $1 \leq i \leq j \leq |\mathcal{T}_{\text{T}}|$. Based on the selected quality level q, a percentage of image blocks in $\mathcal{T}_{\text{T}}$ are selected to be significant, namely $q\,\%$. For a specific $T_i \in \mathcal{T}_{\text{T}}$ we have $T_i \in \mathcal{T}_{\text{sig}}$ iff $i < |\mathcal{T}_{\text{T}}| \cdot \frac{q}{100}$.

While textured image blocks are our main concern, there may be some smooth image blocks that are visually detrimental in $\widetilde{\mathbf{I}}$. Thus, we select a small additional amount of smooth image blocks and include them in the set of significant triangles. Analogously to the case above, we sort the image blocks in $\mathcal{T}_{\text{S}}$ in descending order of the local SSIM. Image blocks with a larger local SSIM value are again prioritized. The amount of significant smooth image blocks in $\mathcal{T}_{\text{sig}}$ is set adaptively after thresholding and quantizing the textured significant triangles with C coefficients, see Section 6.1.4. It is controlled through an additional level of quality $0 \leq q_S \leq 1$. Starting with the first image block in $\mathcal{T}_{\text{S}}$, we threshold and quantize smooth image blocks until $q_S \cdot \text{C}$ further coefficients are selected.

6.1.2 Estimating Optimal Edge Weight

Before constructing the weight matrix $\mathbf{W}$, we estimate the optimal edge weight $w_\mathrm{o} \in (0,1]$ for weakly correlated pixels in a textured image block $T \in \mathcal{T}_\mathrm{T}$. Recall from Section 5.8 that for a model texture signal as in Definition 5.7, the optimal edge weight between weakly correlated pixels is given by

$$w_\mathrm{o} = \frac{\sigma_s^2}{\sigma_g^2 + \sigma_s^2}, \tag{6.1}$$

where σ_s^2, $\sigma_g^2 \in \mathbb{R}^+$ are the variances of the variations in smooth regions and the random gaps at weak transitions respectively.

The design of the model texture signal was chosen to characterize an ideal case, where the locations of smooth and weak transition regions are known deterministically and the statistics of $\varepsilon \sim \mathcal{N}(0, \sigma_s^2)$ and $g \sim \mathcal{N}(\mu_g, \sigma_g^2)$ known probabilistically. In practice however, neither the locations nor statistics of smooth and weak transition regions are known in a given texture signal. Therefore, we need to locate the weak transition regions and then estimate σ_s^2 and σ_g^2.

To this end, we construct an unweighted auxiliary graph $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ with $\mathcal{V}_\mathcal{H} = T$. For the edge set $\mathcal{E}_\mathcal{H}$ we apply the 4-way connectivity template if $T \in \mathcal{T}_\mathrm{C}$ and the 8-way connectivity template if $T \in \mathcal{T}_\mathrm{D}$. We partition the resulting edge set $\mathcal{E}_\mathcal{H}$ into two sets, $\mathcal{E}_\mathcal{H} = \mathcal{E}_\mathcal{S} \cup \mathcal{E}_\mathcal{T}$. Edges in a smooth region are assigned to $\mathcal{E}_\mathcal{S}$, while edges in weak transition regions are assigned to $\mathcal{E}_\mathcal{T}$. In particular, an edge $e = (v_i, v_j) \in \mathcal{E}_\mathcal{H}$ is assigned to a set based on the absolute difference of intensities along them in comparison to a threshold $\delta \in \mathbb{R}^+$, i.e.

$$\begin{aligned} e \in \mathcal{E}_\mathcal{S} &\Leftrightarrow |\mathbf{d}(e)| = |\mathbf{f}(j) - \mathbf{f}(i)| \leq \delta, \\ e \in \mathcal{E}_\mathcal{T} &\Leftrightarrow |\mathbf{d}(e)| = |\mathbf{f}(j) - \mathbf{f}(i)| > \delta, \end{aligned} \tag{6.2}$$

where $\mathbf{d} \in \mathbb{R}^M$ is the vector of intensity differences, obtained via the incidence matrix by $\mathbf{d} = \mathbf{J}\mathbf{f}$. An arbitrary orientation has to be set for each edge, which is inconsequential for the computation of σ_s^2 and σ_g^2.

We assume each smooth variation as an i.i.d. random variable, i.e. $\mathbf{d}(e) \sim \mathcal{N}(\mu_s, \sigma_s^2)$ for all smooth edges $e \in \mathcal{E}_\mathcal{S}$. The variance σ_s^2 is estimated via

$$\sigma_s^2 = \frac{1}{|\mathcal{E}_\mathcal{S}|} \sum_{e \in \mathcal{E}_\mathcal{S}} \left(\mathbf{d}(e) - \mu_s\right)^2, \tag{6.3}$$

see [68], where μ_s is the expected value of the smooth variation. Note that contrary to the model texture signal, we do not necessarily have $\mu_s = 0$. It is estimated by

$$\mu_s = \frac{1}{|\mathcal{E}_\mathcal{S}|} \sum_{e \in \mathcal{E}_\mathcal{S}} \mathbf{d}(e). \tag{6.4}$$

To compute the variance σ_g^2 of the random gap g, recall from Definition 5.7 that for an edge

$e = (v_i, v_j) \in \mathcal{E}_\mathcal{T}$ we assumed

$$\mathbf{d}(e) = \mathbf{f}(j) - \mathbf{f}(i) = (-1)^k g + \varepsilon \Leftrightarrow \mathbf{d}(e) - \varepsilon = (-1)^k g$$

for some $k \in \{0, 1\}$ and smooth variation $\varepsilon \sim \mathcal{N}(\mu_s, \sigma_s^2)$. In order to estimate σ_g^2, we further assume that $\mu_g > 0$ and set k accordingly. The variance is estimated as in (6.3) by

$$\sigma_g^2 = \frac{1}{|\mathcal{E}_\mathcal{T}|} \sum_{e \in \mathcal{E}_\mathcal{T}} \left(|\mathbf{d}(e) - \mu_s| - \mu_g \right)^2, \tag{6.5}$$

where the expected value μ_g is given similarly to (6.4) by

$$\mu_g = \frac{1}{|\mathcal{E}_\mathcal{T}|} \sum_{e \in \mathcal{E}_\mathcal{T}} |\mathbf{d}(e) - \mu_s| .$$

In order to estimate the optimal edge weight w_o, it remains to set the threshold δ. To select δ, we first examine the characteristics of a textured image block. As it displays fine scale details of varying degree, we expect a significantly larger variance over transition edges than on smooth edges. Hence, we assume $\sigma_g^2 \geq 1.5 \cdot \sigma_s^2$ and are thus able to give an upper bound to (6.1):

$$w_\mathrm{o} = \frac{\sigma_s^2}{\sigma_g^2 + \sigma_s^2} \leq 0.4.$$

Additionally, we set a lower bound for the optimal edge weight w_o. Very small values of w_o lead to a disproportional large separation between weakly correlated pixels. This may cause seemingly non-correlated pixels in a single block and leads to a graph Fourier transform with disconnected blocks. Hence, we limit w_o by

$$0.05 \leq w_\mathrm{o} \leq 0.4.$$

Finally, we determine δ via an iterative process. The inital threshold δ^1 is set as the mean of the absolute differences of intensities, i.e.

$$\delta^1 = \frac{1}{M_\mathcal{H}} \sum_{i=1}^{M_\mathcal{H}} |\mathbf{d}(i)| .$$

In any of the iteration steps, w_o^k is computed as in (6.1), with σ_s^2 and σ_g^2 estimated as in (6.3) and (6.5) respectively and $\delta = \delta^k$. If $w_\mathrm{o}^k \geq 0.4$ we set $\delta^{k+1} = 0.9 \cdot \delta^k$ and go to the next iteration step. Otherwise, for $w_\mathrm{o}^k < 0.4$, the iteration terminates and we set $w_\mathrm{o} = \max(w_\mathrm{o}^k, 0.05)$. We remark that an initial value of $w_\mathrm{o}^1 < 0.05$ happens very rarely for natural textured signals, thus only lowering the threshold during the iteration has proven sufficient.

6.1.3 Construction of the Weight matrix

Having computed the optimal edge weight w_o for weakly correlated pixels, we turn to the construction of the weight matrix $\mathbf{W}$. Recall that T is in one of three classes, based on its structure tensor. Class 1 constists of smooth blocks, Class 2 of textured blocks with a dominant principal gradient and Class 3 of blocks with complex textures. Each class has a different construction method for $\mathcal{E}$ and $\mathbf{W}$.

Smooth blocks do not have textures that are pronounced enough to justify the additional overhead cost for a graph description. Thus, the standard graph for image blocks is chosen, which is the 8-way connectivity graph with constant edge weight 1, see Section 5.3.

The textures on image blocks with a dominant principal gradient are efficiently summarized by the principal gradient. Thus, the edges and the weight matrix are constructed based on the graph template of the corresponding group, see Section 5.6. Groups are assigned via the quantized edge orientation $q_{\mathbf{u}_1} \in \{0°, 45°, 90°, 135°\}$ that is most orthogonal to the principal gradient $\mathbf{u}_1$. It is hence selected by

$$q_{\mathbf{u}_1} = \underset{q \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right\}}{\operatorname{argmin}} \left| \langle \mathbf{u}_1, (\cos q, \sin q)^T \rangle \right|.$$

The edge weight w_o for weakly correlated pixels is computed as shown in the previous section. We remark that we do not compute w_o with the graph weights given by the corresponding graph template, since the principal gradient only gives an indication of structure. Even for a large coherence there are irregularities inflating the variances in (6.1).

In contrast, the complex textures on image blocks in Class 3 are not easily summarized. We introduced two methods in Section 5.7 to construct the edge weight vector $\mathbf{w}$. First, a weight vector $\mathbf{w}_c \in \mathbb{R}^M$ with continuous weights was designed to select the optimal GFT for high bit rates. The second approach designed the weight vector $\mathbf{w}_d \in \{w_o, 1\}^M$ with discrete weights based on a threshold $\gamma \in \mathbb{R}^+$. In practice, we apply both methods and select the graph that minimizes the smoothness of $\mathbf{f}$ in regard to it. For both approaches the edge set $\mathcal{E}$ is constructed by the 4-way connectivity template to minimize the amount M of data in $\mathbf{w}$.

In order to optimize the GFT we minimize

$$\begin{aligned} &\min_{\mathbf{w} \in \mathbb{R}^M} \mathbf{s}^T \mathbf{w} + \alpha \left\| \mathbf{U}_\ell^T \mathbf{w} \right\|_{\ell^1} - \beta \mathbf{1}^T \log(\mathbf{w}), \\ &\text{s.t. } \mathbf{w} \leq \mathbf{1} \end{aligned} \tag{6.6}$$

and let the minimizer be $\mathbf{w}_c$. To this end, we first set values for the parameters $\alpha, \beta > 0$. Recall that α weighs the sparseness of $\hat{\mathbf{w}}$ via the ℓ^1-norm, where $\hat{\mathbf{w}}$ are the graph Fourier coefficients of $\mathbf{w}$ in regard to $\mathcal{G}_\ell$. Since we apply the optimization to blocks with complex textures, we expect that consecutive edges have dissimilar intensity differences along them. Therefore, we emphasize the importance of constructing a weight vector $\mathbf{w}$ with compacted values in the graph spectral domain by choosing a large α. In comparison, the penalty for small weights $W_{i,j} \approx 0$,

influenced by the choice of β, is less important. Consequently, we set β significantly smaller than α. For the implementation we chose the values of α and β by fine tuning, letting $\alpha = 10M$ and $\beta = 0.01M$. The optimization problem in (6.6) is a convex constrained minimization problem and solved efficiently via interior-point methods. In the implementation we used the *fmincon* function of the Matlab optimization toolbox. For the initial vector $\mathbf{w}_0$, we employ the Gaussian weight function, so that

$$[\mathbf{w}_0]_e = \exp\left(-\frac{|\mathbf{f}(j) - \mathbf{f}(i)|^2}{\alpha}\right) \tag{6.7}$$

for $e = (i, j) \in \mathcal{E}$, where the Gaussian parameter α is given by

$$\alpha = \frac{1}{3} \cdot \max_{(i,j)\in\mathcal{E}} |\mathbf{f}(j) - \mathbf{f}(i)|^2 .$$

In the second approach $\mathbf{w}_d$, is constructed by fixing a threshold $\gamma \in \mathbb{R}^+$ and assigning either weight 1 or w_o to an edge based on the sign of $\mathbf{d}_\gamma$, see Section 5.7.2. Selecting the threshold γ corresponds to assigning edges weak and strong correlation. It is thus closely related to the computation of the optimal edge weight w_o. Therefore, we set $\gamma = \delta$, where δ is the threshold computed in Section 6.1.2 that assigned weak and strong correlation to edges during the construction of a sensible edge weight w_o.

In order for $\mathbf{w}_c$ and $\mathbf{w}_d$ to be encoded efficiently, recall that they are both graph signals on the line graph $\mathcal{G}_\ell$. Thus, we convert $\mathbf{w} \in \{\mathbf{w}_c, \mathbf{w}_d\}$ to the graph spectral domain via

$$\hat{\mathbf{w}} = \mathbf{U}_\ell^T \mathbf{w}, \tag{6.8}$$

where $\mathbf{L}_\ell = \mathbf{U}_\ell \mathbf{\Lambda}_\ell \mathbf{U}_\ell^T$ is the eigen-decomposition of the graph Laplacian of $\mathcal{G}_\ell$. Due to the choice of large α in (6.6) and the exploitation of the structure of $\mathbf{U}_\ell$ in the second method, we expect the significant characteristics of $\mathbf{w}$ to be compacted in few coefficients. In order to reduce the graph transmission cost, we threshold $\hat{\mathbf{w}}$ and select only the absolute largest coefficients for encoding via a threshold $\varphi_\mathbf{w} \in \mathbb{R}$

$$\hat{\mathbf{w}}_\mathrm{th}(k) = \begin{cases} \hat{\mathbf{w}}(k) & \text{if } |\hat{\mathbf{w}}(k)| \geq \varphi_\mathbf{w}, \\ 0 & \text{if } |\hat{\mathbf{w}}(k)| < \varphi_\mathbf{w} \end{cases} \tag{6.9}$$

for all $1 \leq k \leq M$. In practice, we have observed that very few coefficients are necessary to represent the elemental structure of an edge weight signal sufficiently well. Thus, $\varphi_\mathbf{w}$ is chosen to admit five Fourier coefficients. To this end, we let $|\hat{w}_1| \geq \cdots \geq |\hat{w}_5|$ be the five absolute largest Fourier coefficients and set $\varphi_\mathbf{w} = |\hat{w}_5|$ in the implementation. We reconstruct the edge weight vector via the inverse GFT from the thresholded edge weight vector $\hat{\mathbf{w}}_\mathrm{th}$

$$\mathbf{w} \approx \mathbf{w}_\mathrm{th} = \mathbf{U}_\ell \hat{\mathbf{w}}_\mathrm{th}.$$

For $\mathbf{w} = \mathbf{w}_c$ we exploit the characteristics of the optimization further. First, recall that $\mathbf{w}_c \in (0, 1]^M$. Since this property is not guaranteed for a reconstruction with a sparse $\hat{\mathbf{w}}_\mathrm{th}$, we

rescale $\mathbf{w}_{\text{th}}$ to the acceptable range of weights $[0.05, 1]$ introduced in Section 6.1.2 via

$$\mathbf{w}_{\text{rs}}(k) = 0.95 \cdot \frac{\mathbf{w}_{\text{th}}(k) - \mathbf{w}_{\min}}{\mathbf{w}_{\max} - \mathbf{w}_{\min}} + 0.05 \tag{6.10}$$

for all $1 \leq k \leq N$, where $\mathbf{w}_{\min}$ and $\mathbf{w}_{\max}$ are the minimal and maximal coefficients in $\mathbf{w}_{\text{th}}$ respectively. But due to the rescaling in (6.10) the constant eigenvector $\mathbf{u}_{\ell,0}$ corresponding to $\lambda_0 \in \sigma(\mathbf{L}_\ell)$ is irrelevant to the construction of $\mathbf{w}_{\text{rs}}$. Thus, we set $\hat{\mathbf{w}}(1) = 0$ before thresholding.

Moreover, the choice of a large α may lead to an even more effective compaction of coefficients. Hence, we set two boundaries for $\varphi_{\mathbf{w}}$. First, if a coefficient is small compared to the largest coefficient $\hat{w}_1$, its significance in reconstructing $\mathbf{w}_c$ is negligible. Second, if the relative difference between two consecutive weights is small, the significance in reconstructing $\mathbf{w}_c$ for smaller coefficients is also negligible. In the implementaion we have thus limited $\varphi_{\mathbf{w}}$ via $\varphi_{\mathbf{w}} \geq 0.2 \cdot |\hat{w}_1|$ and $\varphi_{\mathbf{w}} = |\hat{w}_k|$ if $|\hat{w}_k| > 0.4 \cdot |\hat{w}_{k+1}|$ for some $1 \leq k \leq 4$.

Having constructed both $\mathbf{w}_{c,\text{rs}}$ and $\mathbf{w}_{d,\text{th}}$, we select one of them for further processing. To select either $\mathbf{L}_{c,\text{rs}}$ or $\mathbf{L}_{d,\text{th}}$ as the basis for the GFT on a specific block T with complex textures, we evaluate the smoothness of $\mathbf{f}$ in regard to both of them. As seen before, the smoothness of a graph signal depends on the structure of the underlying graph. Since the topology $\mathcal{E}$ is fixed, the smoothness of a graph signal is only affected by the edge weight vectors. We again measure the smoothness of $\mathbf{f}$ in regard to a graph through the graph Laplacian quadratic form, while taking the inherent preference of a lower edge weight into account. Similar to Section 5.7.1 we employ a logarithmic penalty. Hence, we evaluate the smoothness $s(\mathbf{f}, \widetilde{\mathbf{w}})$ of $\mathbf{f}$ in regard to an edge weight vector $\widetilde{\mathbf{w}} \in \{\mathbf{w}_{c,\text{rs}}, \mathbf{w}_{d,\text{th}}\}$ by

$$\begin{aligned} s(\mathbf{f}, \widetilde{\mathbf{w}}) &= \mathbf{f}^T \widetilde{\mathbf{L}} \mathbf{f} - \mathbf{1}^T \log \widetilde{\mathbf{w}} \\ &= \mathbf{f}^T \mathbf{J}^T \text{diag}(\widetilde{\mathbf{w}}) \mathbf{J} \mathbf{f} - \mathbf{1}^T \log(\widetilde{\mathbf{w}}), \end{aligned} \tag{6.11}$$

where $\mathbf{1}$ denotes the constant 1 vector. We compare the smoothness of both edge weight vectors and select the minimizer of (6.11), i.e.

$$\widetilde{\mathbf{w}} = \underset{\mathbf{w} \in \{\mathbf{w}_{c,\text{rs}}, \mathbf{w}_{d,\text{th}}\}}{\text{argmin}} \; s(\mathbf{f}, \mathbf{w}).$$

The corresponding graph Laplacian is then constructed via

$$\widetilde{\mathbf{L}} = \mathbf{J}^T \text{diag}(\widetilde{\mathbf{w}}) \mathbf{J}.$$

6.1.4 Coefficient Thresholding and Quantization

In Section 5.3 we have shown how to convert an image signal $\mathbf{f}$ on a block T to the graph spectral domain. To this end, a graph $\mathcal{G}$ was constructed on the pixels of T and a GFT was defined via the graph Laplacian $\mathbf{L}$ of $\mathcal{G}$

$$\hat{\mathbf{f}} = \mathbf{U}^T \mathbf{f}, \tag{6.12}$$

where $\mathbf{U}$ is the eigenvector matrix of $\mathbf{L}$, defined via the eigen-decomposition $\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$.

Equation (6.12) represents the graph signal $\mathbf{f}$ equivalently in the graph spectral domain. This does not decrease the energy of the signal, see (4.11). In the previous sections we examined how to construct a weight matrix that compacts the energy of the signal. We now exploit the compactness of energy in the graph spectral domain by thresholding small Fourier coefficients. Additionally, we reduce the accuracy of Fourier coefficients to a discrete set through quantization. Our aim for this section is thus twofold: discard information from $\hat{\mathbf{f}}$ that is not visually significant and quantize the remaining data to prepare it for encoding.

First, we consider textured image blocks. In Chapter 5 we described how to construct adaptive graphs $\mathcal{G}$ so that a signal $\mathbf{f}$ on a textured block is smooth with regard to it. Most of the relevant information of $\mathbf{f}$ is therefore contained in a small amount of coefficients in $\hat{\mathbf{f}}$. Hence, we set most of the coefficients 0, based on a threshold $\varphi_{\hat{\mathbf{f}}}$

$$\hat{\mathbf{f}}_{\mathrm{th}}(k) = \begin{cases} \hat{\mathbf{f}}(k) & \text{if } \left|\hat{\mathbf{f}}(k)\right| \geq \varphi_{\hat{\mathbf{f}}}, \\ 0 & \text{if } \left|\hat{\mathbf{f}}(k)\right| < \varphi_{\hat{\mathbf{f}}}. \end{cases} \tag{6.13}$$

The choice for this threshold depends on the characteristics of an image. Keeping the number of total coefficients the same, there is a trade-off between the amount of triangles improved and the detail given on each improved triangle. An image with some large and many small, negligible textured image blocks would benefit from a large $\varphi_{\hat{\mathbf{f}}}$, while the opposite is true for images with many medium sized textured image blocks. The optimal choice of $\varphi_{\hat{\mathbf{f}}}$ is still unclear. In this thesis, we set it to admit 15% of coefficients in $\hat{\mathbf{f}}$, up to a maximum of $10 \cdot q_F$ coefficients. This has proven to result in good SSIM values.

Having thresholded $\hat{\mathbf{f}}$, we employ a uniform quantizer for the quantization of $\hat{\mathbf{f}}_{\mathrm{th}}$. In this thesis, we assume a *fixed rate* for the coeffcients, where all theoretical binary codewords are of equal length. We select four bits, since the construction of $\mathbf{I}_t = \mathbf{I} - \widetilde{\mathbf{I}}$ generates a relatively small range of possible signal values. Thus, the quantization has $2^4 = 16$ levels.

For an optimal result, the quantization support, and thus the step size, is chosen adaptively for each image. This is achieved by first transforming all significant textured image blocks and quantizing them in a second stage. While transforming each textured significant triangular image block, the overall smallest and largest Fourier coefficients are stored and later included in the header for later reconstruction. In particular, we have

$$\begin{aligned} \hat{\mathbf{f}}_{\min} &= \min_{T \in \mathcal{T}_{\mathrm{sig}} \cap \mathcal{T}_{\mathrm{T}}} \min_{1 \leq i \leq |T|} \hat{\mathbf{f}}_T(i), \\ \hat{\mathbf{f}}_{\max} &= \max_{T \in \mathcal{T}_{\mathrm{sig}} \cap \mathcal{T}_{\mathrm{T}}} \max_{1 \leq i \leq |T|} \hat{\mathbf{f}}_T(i). \end{aligned} \tag{6.14}$$

The support of the quantization is subsequently chosen as $[\lfloor \hat{\mathbf{f}}_{\min} \rfloor, \lceil \hat{\mathbf{f}}_{\max} \rceil]$. Consequently, we set the quantization step size to

$$\Delta = \frac{\left\lceil \hat{\mathbf{f}}_{\max} \right\rceil - \left\lfloor \hat{\mathbf{f}}_{\min} \right\rfloor}{15}. \tag{6.15}$$

Thus, the quantization levels are given by

$$y_i = \Delta \cdot \text{Round}\left(\frac{\lfloor \hat{\mathbf{f}}_{\min} \rfloor}{\Delta}\right) + i\Delta$$

for $0 \leq i \leq 15$. From there, we compute the index to be potentially encoded of a Fourier coefficient $\hat{\mathbf{f}}(k)$ via

$$\hat{\mathbf{f}}_{\text{ind}}(k) = \text{Round}\left(\frac{\hat{\mathbf{f}}_{\text{th}}(k)}{\Delta}\right)$$

for $1 \leq k \leq N$. Finally, the complete quantization rule is

$$\hat{\mathbf{f}}_{\text{q}}(k) = \Delta \cdot \text{Round}\left(\frac{\hat{\mathbf{f}}_{\text{th}}(k)}{\Delta}\right). \tag{6.16}$$

Turning to smooth image blocks, we note that no extra structure is introduced to $\mathbf{U}$ through the choice of weights. Hence, the elemental frequencies behave in a way similar to the base frequencies of the conventional DCT, with our usual understanding of visible frequency. Therefore, in order to quantize and threshold $\mathbf{f}$ we take an approach similar to the quantization matrices in the JPEG scheme introduced in Section 2.3. Since $\mathbf{f} \in \mathbb{R}^N$ is given as a vector, we consider *quantization vectors*.

Definition 6.1 *Given a Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ of a graph $\mathcal{G}$, a* quantization vector *is a vector $\mathbf{q} \in \mathbb{R}^N$, where each element $\mathbf{q}(k)$, $1 \leq k \leq N$, represents the quantizer step size of its corresponding frequency $\lambda_{k-1} \in \sigma(\mathbf{L})$.*

Note that the definition given here is unrelated to prototype vectors from vector quantization. Given a quantization vector $\mathbf{q}$, we obtain the usually sparse, quantized signal to be encoded $\hat{\mathbf{f}}_{\text{ind}}$ by dividing each Fourier coefficient by its corresponding quantizer step size and rounding to the nearest integer as in (2.14). In order to minimize the data to be encoded, $\mathbf{q}$ is chosen to only depend on the block size N and be otherwise independent of the signal $\mathbf{f}$. To acquire the quantizer step sizes, we have to determine the psychovisual importance of each base frequency given by the eigenvectors in $\mathbf{U}$. Doing this even for a small set of representative triangles exceeds the scope of this thesis, as the large amount of varying shapes and sizes make psychovisual experiments very cumbersome.

Instead, we build upon the plentiful research conducted on quantizing JPEG images. In (2.15) a standard quantization matrix $\mathbf{Q}_S$ was given for an 8×8 pixel block. Each entry denotes the quantization step size of its corresponding elemental frequency shown in Figure 2.4. Psychovisual experiments have shown that degrading horizontal or vertical textures has a different influence on the perceived quality of an image reconstruction. This is reflected in the standard quantization matrix, where different step sizes are selected for essentially the same textures rotated by 90 degrees.

To generalize the quantization matrix to our setting, we utilize a novel quantization matrix introduced in [31]. It does not treat each entry individually, but sets step sizes for each frequency

order. As the JPEG compression scheme analyzes signals on 8×8 pixel blocks, there are 15 orders of frequency, given by the k-antidiagonals in $\mathbf{Q}_S$ for $-7 \leq k \leq 7$. Transferred to a standard quantization vector, we have

$$\mathbf{q}_S = (16, 14, 13, 15, 19, 28, 37, 55, 64, 83, 103, 117, 117, 111, 90)^T \in \mathbb{R}^{15}, \tag{6.17}$$

where each component in $\mathbf{q}_S$ corresponds to an order of frequency.

While 15 orders of frequency are sufficient for an 8×8 image block, when considering graph signals in the graph spectral domain it generally is not. A graph signal $\mathbf{f} \in \mathbb{R}^N$ with N data points has $|\sigma(\mathbf{L})|$ frequency orders. Large blocks usually have the largest impact on the SSIM, so we mostly have $|\sigma(\mathbf{L})| > 15$. Additionally, the graph frequency orders are not evenly spaced. Thus, the vector given in (6.17) is not adequate for most image blocks in $\mathcal{T}_\mathrm{S}$.

To adapt $\mathbf{q}_S$ to each specific block size we define a linear spline interpolant $h_S : [0,1] \to \mathbb{R}^+$ of $\mathbf{q}_S$, so that $h_S(\frac{k-1}{14}) = \mathbf{q}_S(k)$ for $1 \leq k \leq 15$. Next, we normalize the frequencies orders of $\mathbf{L}$ via $\bar{\lambda}_k = \frac{\lambda_k}{\lambda_{N-1}}$ for $0 \leq k \leq N-1$. Finally, since we know the image blocks to be smooth and only require crucial information to be encoded, we consider a lower level of quantization. Thus, the final quantization vector $\mathbf{q}_T \in \mathbb{R}^N$ of a smooth image block T is given by

$$\mathbf{q}_T(k) = 2.5 \cdot h_S\left(\bar{\lambda}_{k-1}\right)$$

for all $1 \leq k \leq N$.

While the values in (6.17) are suitable for coefficients derived during the JPEG scheme, the values of graph Fourier coefficients tend to be smaller. The range of extremal graph Fourier coefficients in an image is taken from (6.14). As an approximation we assume the DCT coefficients in the JPEG scheme to be in the range $[-128, 128]$. Therefore, in order to threshold and quantize $\hat{\mathbf{f}}$, we compute the coefficients to be encoded via

$$\hat{\mathbf{f}}_\mathrm{ind}(k) = \mathrm{Round}\left(\frac{256}{\left|\lceil \hat{\mathbf{f}}_\mathrm{max} \rceil - \lfloor \hat{\mathbf{f}}_\mathrm{min} \rfloor\right|} \cdot \frac{\hat{\mathbf{f}}(k)}{\mathbf{q}_T(k)}\right)$$

for $1 \leq k \leq N$. The complete quantization rule is given by

$$\hat{\mathbf{f}}_\mathrm{q}(k) = \mathbf{q}_T(k) \frac{\left|\lceil \hat{\mathbf{f}}_\mathrm{max} \rceil - \lfloor \hat{\mathbf{f}}_\mathrm{min} \rfloor\right|}{256} \cdot \hat{\mathbf{f}}_\mathrm{ind}(k). \tag{6.18}$$

Finally, we turn to quantizing the adaptive graph descriptions. To this end, we quantize w_o with a uniform quantizer with step size $\Delta_{w_\mathrm{o}} = 0.05$ and eight levels $\mathcal{L} = \{0.05, 0.10, \ldots, 0.4\}$. Thus, the discrete value of the optimal edge weight is given by $w_{\mathrm{o,q}} \in \{k \cdot 0.05\}_{1 \leq k \leq 8}$ and computed via

$$w_{\mathrm{o,q}} = \left(\left\lfloor \frac{w_\mathrm{o}}{\Delta_{w_\mathrm{o}}} - \frac{1}{2} \right\rfloor + 1\right) \cdot \Delta_{w_\mathrm{o}}.$$

The quantization step size Δ_{w_o} is chosen so that $w_{\mathrm{o,q}}$ can be stored in three bits, while still being an adequate approximation $w_{\mathrm{o,q}} \approx w_\mathrm{o}$.

In order to quantize $\hat{\mathbf{w}}_{\text{th}}$, we distinguish between the continuous and discrete mode. The edge weight coefficient vector of the discrete mode is in the range of the regular coefficients. Therefore, it is quantized as in (6.16). For the continuous mode the range is considerably smaller, as the data to be encoded is in the range $[0.05, 1]$. In practice it is sufficient to constrain $\hat{\mathbf{w}}_{c,\text{rs}} \in [-2, 2]^M$ and uniformly quantize it with $\Delta_{\mathbf{w}} = 0.5$ and eight levels as above, so that

$$\hat{\mathbf{w}}_{\text{ind}}(k) = \left\lfloor \frac{\mathbf{w}_{c,\text{rs}}(k)}{\Delta_{\mathbf{w}}} \right\rfloor,$$

and

$$\hat{\mathbf{w}}_{\text{q}}(k) = \Delta_{\mathbf{w}} \left(\left\lfloor \frac{\mathbf{w}_{c,\text{rs}}(k)}{\Delta_{\mathbf{w}}} \right\rfloor + 0.5 \right),$$

for all $1 \leq k \leq M$.

6.1.5 Algorithm and Image Reconstruction

Finally, we are in a position to fully describe the ATGSP algorithm. The input of the algorithm is given by the original image $\mathbf{I}$, an AT reconstruction $\widetilde{\mathbf{I}}$ and a quality level $1 \leq q \leq 100$. Additionally, an optional smooth quality level $0 \leq q_S \leq 1$ may be provided. If no q_S is given, the standard value $q_S = 0.05$ is set.

First, the texture image $\mathbf{I}_t = \mathbf{I} - \widetilde{\mathbf{I}}$ is set, which is the signal we aim to approximate. Each triangular image block is classified via the structure tensor. Small blocks are automatically classified as smooth, since the graph description overhead would negate any benefits. As explained in Section 6.1.1, we order $\mathcal{T}_{\text{T}}$ and $\mathcal{T}_{\text{S}}$ in descending order according to their local SSIM value.

Next, the adaptive weight matrix of every textured image block in $\mathcal{T}_{\text{sig}}$ is created. We remark that even during the transformation phase the quantized matrices are used. Section 6.1.3 has shown that the overhead for adaptive graphs on image blocks with complex textures is rather large. Therefore, we employ another threshold $\varphi_a \in \mathbb{R}^+$ that sets a minimal block size for complex adaptive graphs. If a block $T \in \mathcal{T}_{\text{C}}$ is small, i.e. $|T| \leq \varphi_a$, it is transformed with a non-adaptive graph, i.e. one that has the standard 8-way connectivity topology with constant edge weight 1. Only if $|T| > \varphi_a$ is an adaptive graph constructed as described in Section 6.1.3. The threshold φ_a is selected based on the quality factor q_F and depends on the image resolution $X_1 \times X_2$. In the implementation it is given by

$$\varphi_a = \left\lfloor \max\left(\frac{0.00075 \cdot X_1 \cdot X_2}{q_F}, 0.0005 \cdot X_1 \cdot X_2, 20 \right) \right\rfloor.$$

Every significant textured block is tranformed via $\hat{\mathbf{f}} = \mathbf{U}^T \mathbf{f}$ and the quantization step size is computed as in (6.15). Afterwards, the Fourier coefficients are thresholded and quantized as in (6.13) and (6.1.4) respectively.

Once all textured blocks $T \in \mathcal{T}_{\text{sig}} \cap \mathcal{T}_{\text{T}}$ are transformed and quantized, we turn to smooth blocks. As discussed in Section 5.2, smooth blocks of $\mathbf{I}_t$ do not contribute as much to the visual quality of an image. Therefore, we only use a small percentage of them, controlled by q_S. In particular, if $\text{C} \in \mathbb{N}^0$ coefficients are used to describe the textured blocks, $q_S \cdot \text{C}$ coefficients are

selected to describe smooth blocks. To this end, we transform smooth image blocks via $\hat{\mathbf{f}} = \mathbf{U}^T\mathbf{f}$ and quantize the coefficients as in (6.1.4), until the desired amount of coefficients to describe smooth blocks is reached. The selection of $T \in \mathcal{T}_\mathrm{S}$ is based on the ordering imposed above.

Algorithm 4: ATGSP

Input: original image $\mathbf{I}$; AT reconstruction $\widetilde{\mathbf{I}}$; triangulation $\mathcal{D}(\mathrm{P}_n)$; quality q; optional smooth quality q_S.

Output: quantized coefficients $\hat{\mathbf{f}}_\mathrm{ind}$; graph description $\hat{\mathbf{w}}_\mathrm{ind}$ or $q_{\mathbf{u}_1}$ and $w_\mathrm{o,ind}$; $\left\lfloor \hat{\mathbf{f}}_\mathrm{min} \right\rfloor$; $\left\lceil \hat{\mathbf{f}}_\mathrm{max} \right\rceil$.

```
Let I_t = I − Ĩ.
Compute classification of every T ∈ D(P_n) based on structure tensor.
Compute priority queue for T_T and T_S.
while k < (q/100) · |T_T| do
    Create Laplacian matrix L_k based on classification of T_k ∈ T_T.
    Compute f̂_k = U_k^T f_k.
Compute quantization step size Δ.
while k < (q/100) · |T_T| do
    Threshold f̂_k.
    Quantize f̂_k based on Δ.
while S < q_S · C do
    Create smooth Laplacian matrix L_k.
    Compute f̂_k = U_k^T f_k.
    Quantize f̂_k.
```

The quantized coefficients $\hat{\mathbf{f}}_\mathrm{ind}$ are output together with the adaptive graph descriptions, i.e. $\hat{\mathbf{w}}_\mathrm{th}$ for complex blocks and $q_{\mathbf{u}_1}$ and $w_\mathrm{o,q}$ for dominant blocks, as well as $\hat{\mathbf{f}}_\mathrm{min}$ and $\hat{\mathbf{f}}_\mathrm{max}$. An overview of the algorithm is displayed in Algorithm 4.

Algorithm 5: Inverse ATGSP

Input: AT reconstruction $\widetilde{\mathbf{I}}$; quantized coefficients $\hat{\mathbf{f}}_\mathrm{ind}$; graph description $\hat{\mathbf{w}}_\mathrm{ind}$ or $q_{\mathbf{u}_1}$ and $w_\mathrm{o,ind}$; $\left\lfloor \hat{\mathbf{f}}_\mathrm{min} \right\rfloor$; $\left\lceil \hat{\mathbf{f}}_\mathrm{max} \right\rceil$.

Output: ATGSP reconstruction $\mathbf{I}_\mathrm{ATGSP}$.

```
Compute quantization step size Δ.
for k = 1, ..., |T_sig| do
    Reconstruct L_k and f_q,k.
    Compute f_q,k = U_k f̂_q,k.
    Add signal f_q,k to Ĩ.
```

To reconstruct the compressed image, first the quantization step size Δ is recomputed. Based on the graph description, the coefficients $\hat{\mathbf{f}}_\mathrm{q}$ are reconstructed as in (6.16) or (6.18). The adaptive graph Laplacians are reconstructed either by selecting the corresponding graph template or via (6.8). Finally, the image block signals are reconstructed via $\mathbf{f}_\mathrm{ATGSP} = \mathbf{U}\hat{\mathbf{f}}_\mathrm{q}$. The final

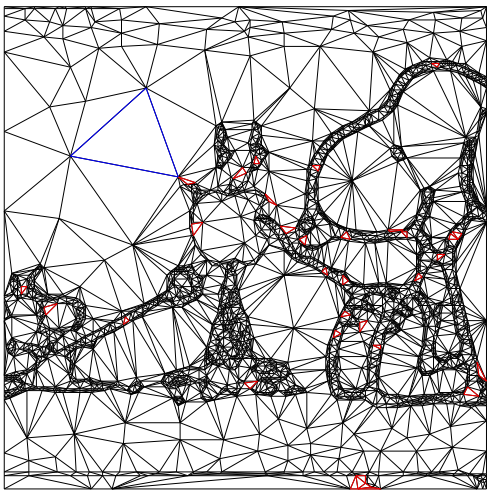
reconstruction $\mathbf{I}_{\mathrm{ATGSP}} \approx \mathbf{I}$ is constructed by converting each graph signal $\mathbf{f}_{\mathrm{ATGSP}}$ back to the pixel setting and adding it to $\widetilde{\mathbf{I}}$. An overview of the process is displayed in Algorithm 5.

6.2 Comparison with Adaptive Thinning on Natural Images

Having described the implementation, we evaluate it on a selection of natural images. As mentioned above, we do not actually encode the quantized coefficients in this thesis. Therefore, we compare our method ATGSP exclusively with the AT compression method in terms of SSIM performance. The basis is the optimal AT reconstruction, i.e. the best approximation over the locally optimal significant pixel set, see Section 3.4.

For a fair comparison, we consider an overhead cost of one byte for every significant image block. Furthermore, even though we quantized all coefficients in Section 6.1.4 to four bits, we consider every coefficient, even those for graph descriptions, to be one byte to account for additional overhead costs. Keep in mind that this is not exact and only a rough estimate.

(a) Original image of Snoopy.



(b) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{3000})$.

(c) AT reconstruction over P_{5000} with SSIM = 0.9845.

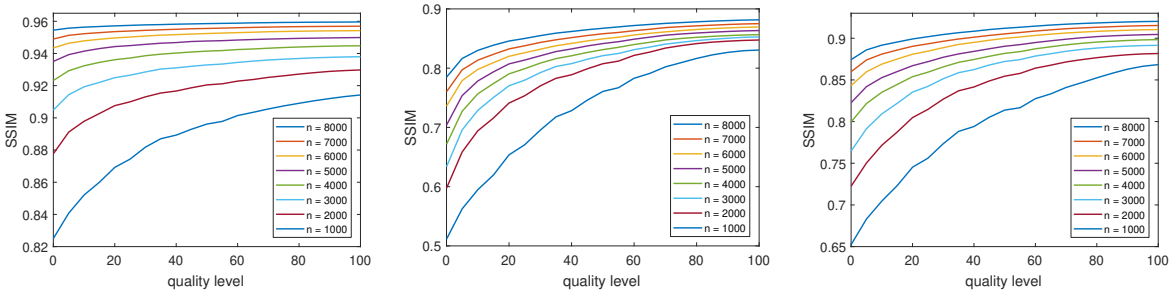
(d) ATGSP reconstruction with 76 coefficients and SSIM = 0.9848.

Figure 6.1: Reconstruction of the image Snoopy via AT and ATGSP.

Every image considered below has a resolution of 256×256 and a bit length of $r = 8$. We consider the standard test images Lena as well as additional test images with varying textural prominence found in [49]. All experiments were performed using the GSPBOX open-source software library [70] in Matlab.

We first compare the two schemes on the image *Snoopy*, displayed in Figure 6.1. The purpose of this comparison is to test the behaviour of ATGSP on images reconstructed very well by AT. In this example, the AT algorithm constructs the Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$ over a set of $n = 3000$ most significant pixels seen in Figure 6.1(b). The AT reconstruction of the image is shown in 6.1(c). Afterwards, we apply the ATGSP algorithm with a quality of $q = 20$ to it. Figure 6.1(b) also displays the significant triangles $\mathcal{T}_{\mathrm{sig}}$. Red triangles are in $\mathcal{T}_{\mathrm{sig}} \cap \mathcal{T}_{\mathrm{T}}$ and blue triangles in $\mathcal{T}_{\mathrm{sig}} \cap \mathcal{T}_{\mathrm{S}}$. ATGSP selects only 76 additional coefficients to reconstruct the signals on those triangles.

Snoopy is a geometrical image, i.e. an image with sharp edges and almost no textures. The AT reconstruction in 6.1(c) is very good, as is evident by a very good visual quality and a SSIM of 0.9845. Thanks to the good AT reconstruction and lack of textured image parts, ATGSP does not select many significant triangles, as is desired. This results in only a marginally better reconstruction with a SSIM of 0.9848. Comparing the two reconstructions with the original image, there is no visible distortion.



(a) Lena. (b) Flower. (c) Pony.

Figure 6.2: Quality-SSIM curves for different n of the images Lena, Flower and Pony.

Having confirmed that the ATGSP post-processing algorithm does not impair a good performance of AT on geometrical images, we now turn to textured images. Before examining specific images in-depth, we consider the quality q in the ATGSP scheme. The ATGSP post-processing scheme adds information for textured image blocks. For constant file sizes, this can be understood as a trade-off between accurately reconstructing geometric features with AT versus reconstructing textures with ATGSP. Hence, there are two possible approaches: (i) improve a small amount of significant triangles on an AT reconstruction with a relatively large amount of significant pixels or (ii) start with an AT reconstruction with less significant pixels and add textures over a large amount of significant triangles. Recall that due to the overhead for an adaptive GFT only $q\%$ of the triangles classified as textured are selected, namely those with large $\mathrm{SSIM}_{\mathrm{local}}$. These

comprise large and heavily textured image blocks. Therefore, we expect most visual information to be already added for a low quality.

To quantify this presumption, we show the SSIM of the representative images Lena, Flower and Pony, reconstructed with ATGSP on different levels of qualities q in Figure 6.2. They were chosen due to their differing textural content. Indeed, we observe vast improvements to the image quality measured via the SSIM for lower qualities. While there are still improvements in the higher quality range, it is not as pronounced. Thus, we adopt approach (i) and select low qualities q.

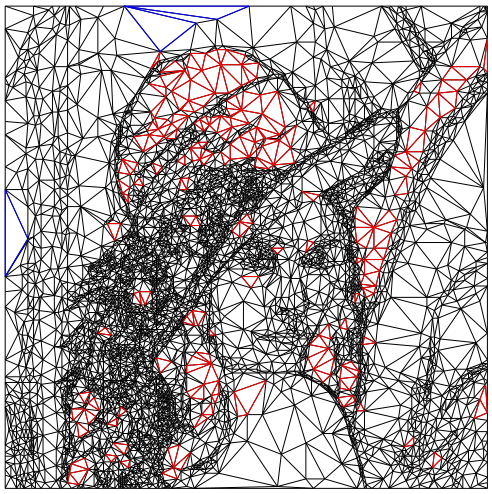
n	AT SSIM	AT filesize	ATGSP SSIM	ATGSP coefficients
1000	0.8250	1586 B	0.8692	1655
2000	0.8777	2847 B	0.9076	1615
3000	0.9049	3992 B	0.9250	1341
4000	0.9232	5040 B	0.9361	1038
5000	0.9351	6037 B	0.9443	854
6000	0.9435	6988 B	0.9497	642
7000	0.9490	7918 B	0.9536	489
8000	0.9545	8789 B	0.9571	333
9000	0.9591	9637 B	0.9608	254
10000	0.9627	10458 B	0.9638	175

Table 6.1: Comparison of AT and ATGSP reconstructions over a selection of n significant pixels with $q = 20$. For each n, the respective SSIM values are given along with the filesize of the AT file and the additional coefficients used in the ATGSP algorithm.

We start with an in-depth comparison of AT and ATGSP on the standard test case *Lena* as an image that comprises a balanced amount of textures and geometries. A quality level of $q = 20$ is used, so that we only add textures to 20% of significant triangles. Next, we have to choose the amount n of significant pixels. To this end, we compare the SSIM of AT and ATGSP for different n in Table 6.1. While the filesize of the AT reconstruction grows steadily, the amount of ATGSP coefficients used declines, along with the improvement by ATGSP. This is as expected, since less significant pixels imply larger triangles with more textural content.

Table 6.1 shows the trade-off between the choices of n. For a small amount of significant pixels the triangles are large, which means much information can be described with few coefficients. But this also leads to a worse approximation of the geometry and more textured triangles. This is observed in the experiment, where the improvement by ATGSP for small n is considerable, but not as good as simply adding more significant pixels in the AT algorithm. On the other hand, for a large amount of significant pixels the image blocks are small, which leads to a very good reconstruction of the geometry and small triangles. This results in very few triangles being considered textured, which means only few additional textures are reconstructed. The size n is chosen to lie inbetween these extrema, where a good performance of ATGSP compared to pure AT is observed.

(a) Original image Lena.



(b) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{5000})$.

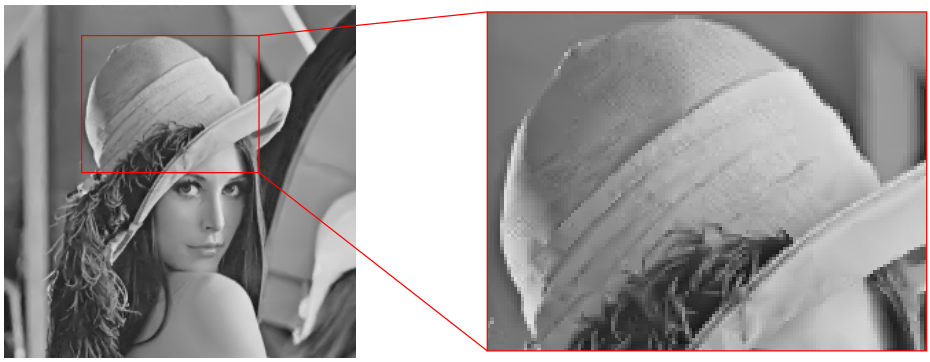
(c) AT reconstruction over P_{6000} with SSIM = 0.9435.

(d) ATGSP reconstruction with 854 coefficients and SSIM = 0.9443.

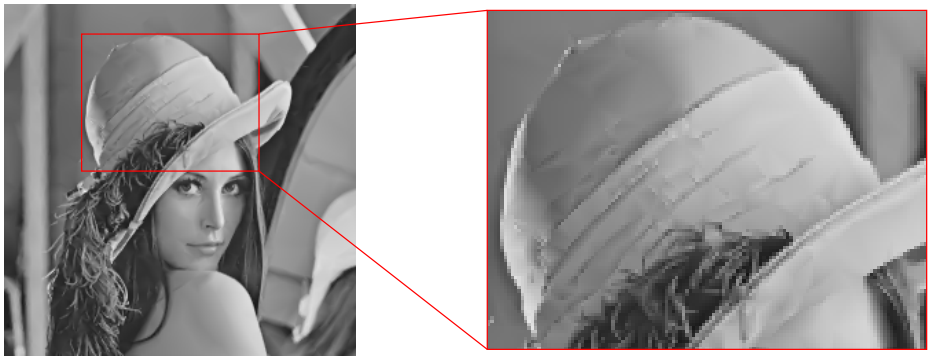
Figure 6.3: Reconstruction of the image Lena via AT and ATGSP.

The original image $\mathbf{I}$ is once more shown in Figure 6.3(a). We reconstruct the image with the ATGSP method over a set of $n = 5000$ significant pixels. The Delaunay triangulation $\mathcal{D}(\mathrm{P}_{5000})$ is shown in 6.3(b). Textured significant triangles are again highlighted in red, while smooth significant triangles are depicted in blue. Note that most of the hat is considered textured, along with parts of her hair, feathers and some of the background structure. This is as expected, since those are the regions we subjectively classified as textured in the discussion in Section 5.2.

We compare the ATGSP reconstruction $\mathbf{I}_{\mathrm{ATGSP}}$ in 6.3(d) with an AT reconstruction $\widetilde{\mathbf{I}}$ over $n = 6000$ significant pixels in 6.3(c). Even though ATGSP uses only 854 coefficients compared to the 951 additional bytes pure AT uses, the SSIM value of $\mathbf{I}_{\mathrm{ATGSP}}$ is larger than that of $\widetilde{\mathbf{I}}$, namely 0.9443 versus 0.9435. Both reconstructions capture the geometry of the image well, while $\mathbf{I}_{\mathrm{ATGSP}}$ has a significantly better portrayal of the textures as seen in Figure 6.4. With the added textures, the ATGSP reconstruction looks more natural than its AT counterpart with flat blocks. While this is a subtle difference, it leads to a better subjective visual quality.



(a) ATGSP reconstruction over $\mathcal{D}(\mathrm{P}_{5000})$ and 854 additional coefficients.



(b) AT reconstruction over $\mathcal{D}(\mathrm{P}_{6000})$.

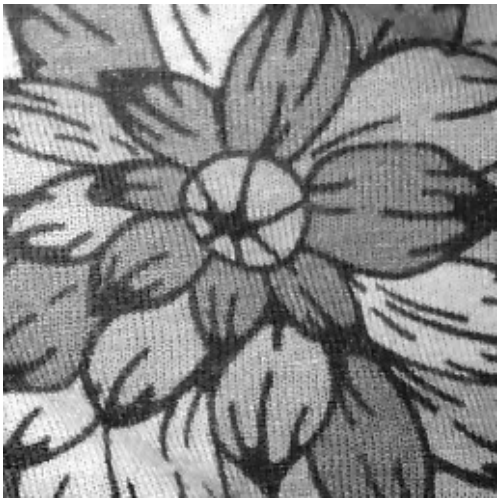
Figure 6.4: Comparison of reconstructed textures on the hat of Lena.

Finally, we turn to compression of mainly textured images, where we consider the textured images *Flower* and *Pony*. As they are both heavily textured, we set $n = 4000$ smaller as above to obtain larger image blocks. Since there are a lot of textures to reconstruct, we set $q_S = 0$ and ignore smooth triangles completely. The reconstruction of heavily structured images with linear splines is intrinsically of poor quality, and triangular artefacts are apparent in both of the following examples.

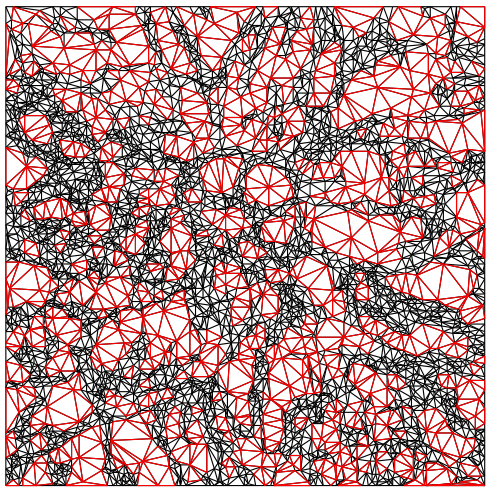
First, we turn to the image Flower in Figure 6.5, where the flower is a clear geometrical feature with a background pattern given by the fabric. In order to capture most large triangles in 6.5(b), we chose $q = 27$ to be relative large. The ATGSP algorithm reconstructs the image with 4399 additional coefficients, so we compare it to the AT reconstruction over P_{8500}. Comparing the SSIM values, we note that the ATGSP reconstruction performs fairly better with a value of 0.8001 compared to 0.7957. In terms of visual quality the flat triangular artefacts are mostly suppressed in 6.5(d).

Next, we turn to the image Pony in Figure 6.6. We set $q = 10$, which gives us a good representation of large, textured triangles in 6.6(b). As above, the ATGSP algorithm with 1573 additional coefficients performs better than its comparable AT reconstruction over P_{5500} with a SSIM value of 0.8346 versus 0.8324. While both reconstructions exhibit triangular artefacts, they are less pronounced in 6.6(d). Especially the mother pony in the background is fully covered by significant triangles and thus reconstructed well.

Our final conclusion is that the ATGSP post-processing algorithm is a good addition to the AT adaptive coding scheme, even in its very early developement stage. It does not impair the good performance of it on geometrical images and improves the reconstruction of textured images noticeably.



(a) Original image Flower.



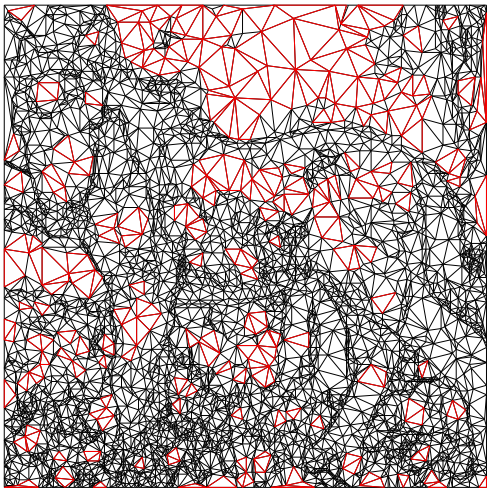
(b) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{4000})$.

(c) AT reconstruction over P_{8500} with SSIM = 0.7957.

(d) ATGSP reconstruction with 4399 coefficients and SSIM = 0.8001.

Figure 6.5: Reconstruction of the image Flower via AT and ATGSP.

(a) Original image Pony.



(b) Delaunay triangulation $\mathcal{D}(\mathrm{P}_{4000})$.

(c) AT reconstruction over P_{5500} with SSIM = 0.8324.

(d) ATGSP reconstruction with 1573 coefficients and SSIM = 0.8346.

Figure 6.6: Reconstruction of the image Pony via AT and ATGSP.

Chapter 7

Summary and Outlook

This thesis was concerned with the compression of digital images via the proposed compression scheme ATGSP. It improved on the AT compression scheme by adding textures on significant triangles.

After providing a brief overview of conventional image compression techniques, we introduced the AT algorithm as a means to adaptively approximate a given digital image $\mathbf{I}$. It selects a sparse set of most significant pixels P_n, over which the ansisotropic Delaunay triangulation $\mathcal{D}(\mathrm{P}_n)$ is generated. The reconstruction $\widehat{\mathbf{I}}$ is then given by a linear spline, defined by optimal luminances of the pixels in P_n. While the representation of geometric features is very good, we have shown that linear splines are inherently unsuitable to reconstruct textures.

We therefore turned to graph signal processing to balance the good reconstruction of geometries by AT with a good representation of textures in the graph spectral domain. GSP defines frequency spectra on irregular domain, such as triangular image blocks. It introduces graph signals $\mathbf{f} \in \mathbb{R}^N$ defined on weighted graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. Based on the graph Laplacian $\mathbf{L}$, a graph Fourier transform was defined that transfers $\mathbf{f}$ to the graph spectral domain via $\hat{\mathbf{f}} = \mathbf{U}^T\mathbf{f}$.

Combining AT and GSP, we proposed our post-processing scheme ATGSP. Additional information of the textural part $\mathbf{I}_t$ of an image is added over a set of significant triangles $\mathcal{T}_{\text{sig}}$. We have shown that it is sensible to define $\mathbf{I}_t$ as the difference between the original image and the AT reconstruction, i.e. $\mathbf{I}_t = \mathbf{I} - \widehat{\mathbf{I}}$.

Each triangular image block $T \in \mathcal{T}_{\text{sig}}$ was converted to the graph setting, by identifying the pixels with vertices, which inherently defined the graph signal $\mathbf{f}$. More care had to be given to the construction of $\mathcal{E}$ and $\mathbf{W}$ in order to construct graphs that exploit the smoothness of $\mathbf{f}$. To this end, we introduced the block structure tensor and classified the significant triangles based on their textures. For each class a graph was constructed that imposed sparseness in the graph spectral domain.

For smooth image blocks without textures, the 8-way connectivity graph was chosen. On anisotropically textured blocks, the graph was constructed via a graph template that was based on the dominant principal gradient $\mathbf{u}_1$. Isotropic textured blocks required more care. A Graph learning approach was compared with a discrete weight model based on the smoothness of $\mathbf{f}$ in regard to them. For both textured classes an optimal edge weight w_{o} was derived, so that the resulting GFT approximates the KLT.

Finally, we described the experimental setup, including thresholding and quantization. Based on this implementation, we compared our proposed ATGSP method to the unaltered AT compression scheme on natural images.

While the numerical experiments on natural images in Section 6.2 show a promising reconstruction on textured images, the ATGSP scheme is still in a very early stage and there is room for further improvement.

Comparing the reconstructions of different images, especially the textures on Lenas hat and on the mother pony were reconstructed very well. This is caused by the relatively large triangles that cover those regions in the Delaunay triangulation $\mathcal{D}$, which results in a small amount of overhead being necessary for a good texture reconstruction. The selection of significant triangles as in Section 6.1.1 is completely dependent on $\mathcal{D}$ output by the AT algorithm. Due to the choice of significance in the AT algorithm, it will cover heavily textured ares with small triangles, which is detrimental to our proposed coding scheme.

In future work, the first step is to either include the texturedness of an area in the AT algorithm or perform further pixel exchanges to combine similarly textured image blocks. While covering uniformly textured regions with large triangles is detrimental to the regular AT coding scheme, together with ATGSP it would improve the representation of textures considerably. Removing pixels from textured regions, it also increases the concentration of significant pixels around edges, leading to a better geometry representation. This should be paired with an efficient encoding of coefficients and graph descriptions integrated into the AT coding scheme.

But while large textured blocks are beneficial for the performance of ATGSP in a quality sense, the computational cost is quite high. The eigenvalue decomposition of $\mathbf{L} \in \mathbb{R}^{N \times N}$ has a worst case computational cost of $\mathcal{O}(N^3)$. Even though only local blocks are chosen and the computation of every block is independent, so that the algorithm is parallelizable, it is still quite expensive. This is especially detrimental compared to algorithms such as JPEG or JPEG 2000 utilizing the fast Fourier transform. The computational cost may possibly be improved by exploiting the known structure of $\mathbf{L}$ or computing approximating eigenvectors. Additionally, it may possibly be advantageous to utilize the non-normalized graph Laplacian $\widehat{\mathbf{L}}$. Due to the absence of a constant eigenvector and $\sigma(\widehat{\mathbf{L}}) \subset [0, 2]$, it may benefit the performance and computational cost.

Next, we discussed that there is no general insight on specific elemental frequencies given the wide variety of shapes and sizes of triangles. Therefore, improving the selection and thresholding methods in Section 6.1 should increase the performance of ATGSP. Especially the thresholding of $\hat{\mathbf{f}}$ would profit from an adaptive selection depending on the elemental frequencies. Furthermore, the *optimal* values of q and q_s depend on the image. Setting these automatically would be a considerable improvement.

Finally, the ATGSP algorithm can be generalized to color images and videos. A popular colour domain for colour images is the *YCbCr* domain, where the Y channel contains the luminance information while the other two contain the chrominance information. They are usually correlated, which could be expoloited by the ATGSP method. Videos can be considered as a sequence of images. The AT scheme applied to them constructs a Delaunay tetrahedralization over the domain, based on a similar construction as in Chapter 3. It might be worthwile to adapt the ATGSP algorithm, especially the perception of textures, to improve the reconstructed videos.

Bibliography

[1] M. Abdelrazek. Image compression using dct upon various quantization. *International Journal of Computer Applications*, 137:11–13, 03 2016.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[3] A. Ahumada and H. Peterso. Luminance-model-based dct quantization for color image compression. *Proc SPIE Human Vision, Visual Process Display III*, 1666, 12 1997.

[4] T. Biyikoglu, J. Leydold, and P. F. Stadler. *Laplacian eigenvectors of graphs: Perron-Frobenius and Faber-Krahn type theorems*. Springer, 2007.

[5] S. P. . Boyd. *Convex optimization*. Cambridge Univ. Press, 18. print. edition, 2015.

[6] E. J. Candes and D. L. Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, Stanford Univ Ca Dept of Statistics, 2000.

[7] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic. Signal denoising on graphs via graph filtering. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 872–876, 2014.

[8] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng. Graph spectral image processing. *Proceedings of the IEEE*, 106(5):907–930, 2018.

[9] C. Christopoulos, A. Skodras, and T. Ebrahimi. The jpeg2000 still image coding system: An overview. *Consumer Electronics, IEEE Transactions on*, 46:1103 – 1127, 12 2000.

[10] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[11] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[12] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

[13] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry*. Springer, third edition, 2008.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[15] L. Demaret and A. Iske. Scattered data coding in digital image compression. *Curve and Surface Fitting: Saint-Malo 2002*, pages 107 – 117, 2003.

[16] L. Demaret and A. Iske. Advances in digital image compression by adaptive thinning. *Annals of the MCFA*, 3:105–109, 2004.

[17] L. Demaret and A. Iske. Adaptive image approximation by linear splines over locally optimal delaunay triangulations. *IEEE Signal Processing Letters*, 13(5):281–284, 2006.

[18] L. Demaret and A. Iske. Anisotropic triangulation methods in adaptive image approximation. In E. H. Georgoulis, A. Iske, and J. Levesley, editors, *Approximation Algorithms for Complex Systems*, pages 47–68, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[19] L. Demaret and A. Iske. Optimally sparse image approximation by adaptive linear splines over anisotropic triangulations. In *2015 International Conference on Sampling Theory and Applications (SampTA)*, pages 463–467, 2015.

[20] L. Demaret, N. Dyn, M. S. Floater, and A. Iske. Adaptive thinning for terrain modelling and image compression. In *Advances in multiresolution for geometric modelling*, pages 319–338. Springer, 2005.

[21] L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[22] L. Demaret, A. Iske, and W. Khachabi. Contextual Image Compression from Adaptive Sparse Data Representations. In R. Gribonval, editor, *SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations*, Saint Malo, France, April 2009. Inria Rennes - Bretagne Atlantique.

[23] L. Demaret, A. Iske, and W. Khachabi. *Sparse Representation of Video Data by Adaptive Tetrahedralizations*, pages 101–121. Springer London, London, 2012. doi: 10.1007/978-1-4471-2353-8_6. URL `https://doi.org/10.1007/978-1-4471-2353-8_6`.

[24] M. N. Do and M. Vetterli. Contourlets: a directional multiresolution image representation. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–I. IEEE, 2002.

[25] D. L. Donoho et al. Wedgelets: Nearly minimax estimation of edges. *Annals of statistics*, 27 (3):859–897, 1999.

[26] N. Dyn, M. S. Floater, and A. Iske. Adaptive thinning for bivariate scattered data. *Journal of Computational and Applied Mathematics*, 145(2):505 – 517, 2002.

[27] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics (tog)*, 9(1):66–104, 1990.

[28] M. Effros, H. Feng, and K. Zeger. Suboptimality of the karhunen-loeve transform for transform coding. *IEEE Transactions on Information Theory*, 50(8):1605–1619, 2004.

[29] H. E. Egilmez and A. Ortega. Spectral anomaly detection using graph-based filtering for wireless sensor networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1085–1089, 2014.

[30] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing. *IEEE Transactions on Image Processing*, 17(7):1047–1060, 2008.

[31] F. Ernawan and S. H. Nugraini. The optimal quantization matrices for jpeg image compression from psychovisual threshold. *Journal of Theoretical and Applied Information Technology*, 70(3):566–572, 2014.

[32] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[33] G. Fracastoro, D. Thanou, and P. Frossard. Graph transform optimization with application to image compression. *IEEE Transactions on Image Processing*, 29:419–432, 2020.

[34] A. Gadde, S. K. Narang, and A. Ortega. Bilateral filter: Graph spectral interpretation and extensions. In *2013 IEEE International Conference on Image Processing*, pages 1222–1226, 2013.

[35] R. Gonzalez. *Digital Image Processing 2Nd Ed.* Prentice-Hall Of India Pvt. Limited, 2002.

[36] C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and compression of 3d meshes. In *Tutorials on Multiresolution in Geometric Modelling*, pages 319–361. Springer, 2002.

[37] B. Goyal, A. Dogra, S. Agrawal, B. Sohi, and A. Sharma. Image denoising review: From classical to state-of-the-art approaches. *Information Fusion*, 55:220–244, 2020.

[38] V. K. Goyal, J. Zhuang, and M. Veiterli. Transform coding with backward adaptive updates. *IEEE Transactions on Information Theory*, 46(4):1623–1633, 2000.

[39] L. J. Grady. *Discrete Calculus Applied Analysis on Graphs for Computational Science*. Springer-Verlag London Limited, 2010.

[40] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998.

[41] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[42] S. Heyman. Photos, photos everywhere, July 2015. URL `https://www.nytimes.com/2015/07/23/arts/international/photos-photos-everywhere.html`.

[43] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[44] W. Hu, X. Li, G. Cheung, and O. Au. Depth map denoising using graph-based transform and group sparsity. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 001–006, 2013.

[45] W. Hu, G. Cheung, A. Ortega, and O. C. Au. Multiresolution graph fourier transform for compression of piecewise smooth images. *IEEE Transactions on Image Processing*, 24(1): 419–433, 2014.

[46] W. Huang, L. Goldsberry, N. F. Wymbs, S. T. Grafton, D. S. Bassett, and A. Ribeiro. Graph frequency analysis of brain signals. *IEEE Journal of Selected Topics in Signal Processing*, 10 (7):1189–1203, 2016.

[47] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[48] A. Iske. *Approximation Theory and Algorithms for Data Analysis.* Springer, 2018.

[49] A. Iske. 100 testbilder zur bildverarbeitung. *Universität Hamburg*, September 2020.

[50] A. Iske and N. Wagner. From image to video approximation by adaptive splines over tetrahedralizations. *Sampling Theory in signal ind Image Processing*, 17:43 – 55, 2018.

[51] B. Jähne. *Spatio-temporal image processing: theory and scientific applications*, volume 751. Springer Science & Business Media, 1993.

[52] A. K. Jain. *Fundamentals of digital image processing.* Englewood Cliffs, NJ: Prentice Hall„ 1989.

[53] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[54] V. Kalofolias. How to learn a graph from smooth signals. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 920–929, Cadiz, Spain, 09–11 May 2016. PMLR.

[55] K. Karhunen. *Über lineare Methoden in der Wahrscheinlichkeitsrechnung*, volume 37. Sana, 1947.

[56] W. Kim, S. K. Narang, and A. Ortega. Graph based transforms for depth video coding. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 813–816, 2012.

[57] D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss. Sparse multidimensional representation using shearlets. In *Wavelets XI*, volume 5914, page 59140U. International Society for Optics and Photonics, 2005.

[58] E. Le Pennec and S. Mallat. Sparse geometric image representations with bandelets. *IEEE transactions on image processing*, 14(4):423–438, 2005.

[59] F. Li and M. K. Ng. Image colorization by using graph bi-laplacian. *Advances in Computational Mathematics*, 45(3):1521–1549, 2019.

[60] M. Loéve. Aléatoires de second order. *Processus Stochastiques et Moevement Brownien*, 1948.

[61] J. Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

[62] S. Mallat and F. Falzon. Analysis of low bit rate image transform coding. *IEEE Transactions on Signal Processing*, 46(4):1027–1042, 1998.

[63] S. G. Mallat. Multiresolution approximations and wavelet orthonormal bases of L^2 (R). *Transactions of the American mathematical society*, 315(1):69–87, 1989.

[64] B. Mohar. Some applications of laplace eigenvalues of graphs. In *Graph symmetry*, pages 225–275. Springer, 1997.

[65] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991.

[66] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka. Graph signal denoising via trilateral filter on graph spectral domain. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):137–148, 2016.

[67] A. V. Oppenheim, J. R. Buck, and R. W. Schafer. *Discrete-time signal processing. Vol. 2.* Upper Saddle River, NJ: Prentice Hall, 2001.

[68] K. I. Park and Park. *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer, 2018.

[69] E. Pavez, H. E. Egilmez, Y. Wang, and A. Ortega. Gtt: Graph template transforms with applications to image coding. In *2015 Picture Coding Symposium (PCS)*, pages 199–203, 2015.

[70] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. Gspbox: A toolbox for signal processing on graphs, 2016.

[71] G. Plonka, S. Tenorth, and A. Iske. Optimally sparse image representation by the easy path wavelet transform. *International Journal of Wavelets, Multiresolution and Information Processing*, 10(01):1250007, 2012.

[72] G. Plonka, A. Iske, and S. Tenorth. Optimal representation of piecewise hölder smooth bivariate functions by the easy path wavelet transform. *Journal of Approximation Theory*, 176:42–67, 2013.

[73] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press Professional, Inc., USA, 1990.

[74] K. R. Rao and P. C. Yip. *The transform and data compression handbook*. CRC press, 2018.

[75] I. Rotondo, G. Cheung, A. Ortega, and H. E. Egilmez. Designing sparse graphs via structure tensor for block transform coding of images. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 571–574, 2015.

[76] I. Rotondo, G. Cheung, A. Ortega, and H. E. Egilmez. Designing sparse graphs via structure tensor for block transform coding of images. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 571–574. IEEE, 2015.

[77] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[78] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27 (3):379–423, 1948.

[79] G. Shen, W. . Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey. Edge-adaptive transforms for efficient depth map coding. In *28th Picture Coding Symposium*, pages 566–569, 2010.

[80] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[81] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[82] J. W. Soh, H. Lee, and N. I. Cho. An image compression algorithm based on the karhunen loève transform. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1436–1439, 2017.

[83] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, man, and cybernetics*, 8(6):460–473, 1978.

[84] D. Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on image processing*, 9(7):1158–1170, 2000.

[85] D. Taubman and M. Marcellin. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated, 2013.

[86] D. Thanou, P. A. Chou, and P. Frossard. Graph-based compression of dynamic 3d point cloud sequences. *IEEE Transactions on Image Processing*, 25(4):1765–1778, 2016.

[87] D. Tian, H. Mansour, A. Knyazev, and A. Vetro. Chebyshev and conjugate gradient filters for graph image denoising. In *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–6, 2014.

[88] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[89] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[90] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[91] S. Yagyu, A. Sakiyama, and Y. Tanaka. Pyramidal image representation with deformation: Reformulation of domain transform and filter designs. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3608–3612, 2016.

[92] K. Yamamoto, M. Onuki, and Y. Tanaka. Deblurring of point cloud attributes in graph spectral domain. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1559–1563, 2016.

[93] J. Yuan, E. Bae, X.-C. Tai, and Y. Boykov. A continuous max-flow approach to potts model. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 379–392, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[94] C. Zhang and D. Florêncio. Analyzing the optimality of predictive transform coding using graph-based models. *IEEE Signal Processing Letters*, 20(1):106–109, 2012.

[95] D. Zhou and B. Schölkopf. Regularization on discrete spaces. In W. G. Kropatsch, R. Sablatnig, and A. Hanbury, editors, *Pattern Recognition*, pages 361–368, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[96] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

List of Symbols

Abbreviations

AT	adaptive thinning	23
DCT	discrete cosine transform	14
DFT	discrete Fourier transform	8
DWT	discrete wavelet transform	18
GFT	graph Fourier transform	42
GSP	graph signal processing	33
JPEG	Joint Photographic Experts Group	2
KLT	Karhunen-Loève transform	5
MSE	mean square error	10
PSNR	peak signal-to-noise ratio	10
SSIM	structural similarity index measure	10
ST	block structure tensor	58

Greek letters

Δ	quantization step size	8
μ	expected value of a random variable	12
$\boldsymbol{\mu}$	expected value of a random signal	12
σ^2	variance of a random variable	72
$\boldsymbol{\Sigma}$	covariance matrix of a random signal	12
$\boldsymbol{\Lambda}$	eigenvalue matrix	35
$\sigma(\mathbf{L})$	spectrum of Ł	42
φ_λ	threshold for largest ST eigenvalue	82
φ_c	threshold for coherence	82
$\varphi_{\mathbf{w}}$	threshold for edge weight vector	87
$\varphi_{\hat{\mathbf{f}}}$	threshold for Fourier coefficients	89
φ_a	threshold for minimum size of complex adaptive graph	92

Indices

ind	index to be encoded	15
th	thresholded	87
q	quantized coefficients	90
ℓ	line graph	38

Latin letters

A Zusammenfassung

Aufgrund der kontinuierlichen Entwicklung von Smarphones mit integrierten Kameras ist die Bildkompression von digitalen Bildern nach wie vor ein relevantes Forschungsgebiet. Herkömmliche Komprimierungsmethoden wie JPEG oder JPEG 2000 basieren auf einer Transformationskodierung über gleichmäßige zweidimensionale Gitter. Im Gegensatz dazu komprimiert die Adaptive Thinning Methode Bilder, indem sie sie mit linearen Splines über der anisotropen Delaunay Triangulierung einer kleinen Menge adaptiv gewählter, signifikanten Bildpunkte approximiert.

Das Hauptziel dieser Arbeit ist die Verbesserung der Adaptive Thinning Methode, insbesondere für texturiert Bilder. Zu diesem Zweck stellen wir einen Nachbearbeitungs-Algorithmus vor, der ausgewählte Regionen eines approximierten Bildes verbessert, indem die graphenbasierte Signalverarbeitung angewendet wird, um eine Fourier Transformation auf unregelmäßigen Strukturen zu definieren.

Unsere Methode konstruiert adaptive Graphen um die Glattheit von Bildsignalen auszunutzen. Hierfür werden signifikante dreieckige Bildblöcke mithilfe des Strukturtensors aufgrund ihres Texturgehalts klassifiziert. Für jede Klasse wird ein Graph konstruiert, der die Spärlichkeit im graphischen Frequenzbereich begünstigt. Dazu wird entweder versucht einen optimalen Graphen zu erlernen oder ein diskretes Gewichtungmodel angewendet, bei dem die Gewichte optimal gewählt sind.

Basierend auf den konstruierten Graphen wird das Signal mit der graphenbasierten Fourier Transformation in den graphischen Frequenzbereich übertragen, wo irrelevante Information entfernt, und die Koeffizienten anschließend quantisiert werden.

Abschließend stellen wir die praktische Implementierung dieser Methode vor, wobei wir uns auf die Transformation und Quantisierung konzentrieren. Wir vergleichen unsere mit der Adaptive Thinning Methode auf geometrischen und texturierten Bildern.

B Publications derived from this dissertation

A. Iske and N. Wagner:

Sparse approximation of videos by adaptive linear splines over anisotropic tetrahedralizations IEEE International Conference *Sampling Theory and Applications (SampTA2017)*, 2017, 251-255.

A. Iske and N. Wagner:

From Image to Video Approximation by Adaptive Splines over Tetrahedralizations *Sampling Theory in Signal and Image Processing* **17**, 2018, 43-55.

C Eidesstattliche Versicherung / Declaration on oath

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare upon oath that I have written the present dissertation independently and have not used further resources and aids than those stated.

Ort, Datum | *city, date*	Unterschrift | *signature*