

# **PDE-based Image Compression Based on Edges and Optimal Data**

Dissertation zur Erlangung des Grades des  
Doktors der Ingenieurwissenschaften  
der Naturwissenschaftlich-Technischen Fakultäten  
der Universität des Saarlandes

vorgelegt von  
**Markus Mainberger**

Saarbrücken  
2014

Tag des Kolloquiums: 06.10.2014

Dekan: Prof. Dr. Markus Bläser  
Universität des Saarlandes

Prüfungsausschuss: Prof. Dr. Holger Hermanns  
Universität des Saarlandes (Vorsitz)

Prof. Dr. Joachim Weickert  
Universität des Saarlandes (1. Gutachter)

Prof. Dr. Søren Forchhammer  
Dänemarks Technische Universität (2. Gutachter)

Prof. Dr. Bernhard Burgeth  
Universität des Saarlandes (akademischer Beisitzer)

---

# Contents

<b>Contents</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.1.1 PDE-based Image Compression . . . . .	2
1.1.2 Objective . . . . .	4
1.1.3 Related Work . . . . .	5
1.2 Outline . . . . .	7
<b>2 PDE-based Inpainting</b>	<b>9</b>
2.1 Motivation . . . . .	9
2.2 Homogeneous Diffusion . . . . .	10
2.3 Homogeneous Diffusion Inpainting . . . . .	11
2.4 Discrete Theory . . . . .	12
2.4.1 Extended Discrete Formulation . . . . .	13
2.4.2 Reduced Discrete Formulation . . . . .	14
2.4.3 Discrete Well-Posedness . . . . .	15
2.5 Efficient Numerical Solvers . . . . .	17
2.5.1 Basic Solver . . . . .	18
2.5.2 Bidirectional Multigrid . . . . .	19
2.5.3 Advanced Multigrid Strategies . . . . .	21
2.5.4 Transfer between Grids . . . . .	22
2.6 Advanced Differential Operators . . . . .	23
2.6.1 Biharmonic Operator . . . . .	24
2.6.2 Edge-Enhancing Anisotropic Diffusion . . . . .	28

2.7	Colour Images . . . . .	31
<b>3</b>	<b>Edge-based Image Compression</b>	<b>33</b>
3.1	Motivation . . . . .	33
3.2	Encoding . . . . .	36
3.2.1	Edge Detection . . . . .	37
3.2.2	Encoding the Contour Location . . . . .	38
3.2.3	Encoding the Contour Pixel Values . . . . .	39
3.2.4	File Format . . . . .	46
3.3	Decoding . . . . .	48
3.4	Experiments . . . . .	49
3.4.1	Comparison with JPEG and JPEG2000 . . . . .	49
3.4.2	Limitations . . . . .	55
3.5	Conclusion . . . . .	56
<b>4</b>	<b>Optimising Spatial and Tonal Data</b>	<b>57</b>
4.1	Motivation . . . . .	57
4.2	Optimising Spatial Data . . . . .	60
4.2.1	The Analytic Approach . . . . .	60
4.2.2	Probabilistic Sparsification . . . . .	63
4.2.3	Non-local Pixel Exchange . . . . .	66
4.3	Optimising Tonal Data . . . . .	71
4.3.1	Least Squares Approximation . . . . .	71
4.3.2	Iterative Approach . . . . .	72
4.3.3	Results . . . . .	75
4.4	Extensions to the Biharmonic Operator and Edge-Enhancing Diffusion . . . . .	83
4.4.1	Spatial Optimisation . . . . .	83
4.4.2	Greyvalue Optimisation . . . . .	85
4.5	Compression Result . . . . .	105
4.6	Conclusion . . . . .	108
<b>5</b>	<b>Edge-based Image Compression revisited</b>	<b>109</b>
5.1	Motivation . . . . .	109
5.2	Integrating Optimal Tonal Data . . . . .	109
5.2.1	Simplified Algorithm . . . . .	109
5.2.2	Results . . . . .	111
5.3	Integrating Optimised Spatial Data . . . . .	115
5.3.1	Hybrid Method . . . . .	117
5.3.2	Results . . . . .	117
5.4	Conclusion . . . . .	122

<b>6 Summary and Outlook</b>	<b>123</b>
6.1 Summary . . . . .	123
6.2 Conclusion and Outlook . . . . .	124
<b>Bibliography</b>	<b>129</b>



---

## Short Abstract

This thesis investigates image compression with partial differential equations (PDEs) based on edges and optimal data.

It first presents a lossy compression method for cartoon-like images. Edges together with some adjacent pixel values are extracted and encoded. During decoding, information not covered by this data is reconstructed by PDE-based inpainting with homogeneous diffusion. The result is a compression codec based on perceptual meaningful image features which is able to outperform *JPEG* and *JPEG2000*.

In contrast, the second part of the thesis focuses on the optimal selection of inpainting data. The proposed methods allow to recover a general image from only 4% of all pixels almost perfectly, even with homogeneous diffusion inpainting. A simple conceptual encoding shows the potential of an optimal data selection for image compression: The results beat the quality of *JPEG2000* when anisotropic diffusion is used for inpainting.

Finally, the thesis shows that the combination of the concepts allows for further improvements.





---

## Kurzzusammenfassung

Die vorliegende Arbeit untersucht die Bildkompression mit partiellen Differentialgleichungen (PDEs), basierend auf Kanten und optimalen Daten.

Sie stellt zunächst ein verlustbehaftetes Kompressionsverfahren für cartoonartige Bilder vor. Dazu werden Kanten zusammen mit einigen benachbarten Pixelwerten extrahiert und anschließend kodiert. Während der Dekodierung, werden Informationen, die durch die gespeicherten Daten nicht abgedeckt sind, mittels PDE-basiertem Inpainting mit homogener Diffusion rekonstruiert. Das Ergebnis ist ein Kompressionscodec, der auf visuell bedeutsamen Bildmerkmalen basiert und in der Lage ist, die Qualität von *JPEG* und *JPEG2000* zu übertreffen.

Im Gegensatz dazu konzentriert sich der zweite Teil der Arbeit auf die optimale Auswahl von Inpaintingdaten. Die vorgeschlagenen Methoden ermöglichen es, ein gewöhnliches Bild aus nur 4% aller Pixel nahezu perfekt wiederherzustellen, selbst mit homogenem Diffusionsinpainting. Eine einfache konzeptuelle Kodierung zeigt das Potential einer optimierten Datenauswahl auf: Die Ergebnisse übersteigen die Qualität von *JPEG2000*, sofern das Inpainting mit einem anisotropen Diffusionsprozess erfolgt. Schließlich zeigt die Arbeit, dass weitere Verbesserungen durch die Kombination der Konzepte erreicht werden können.



---

## Abstract

This thesis investigates image compression with partial differential equations (PDEs) based on edges and optimal data.

It first presents a lossy compression method for cartoon-like images, which is based on edge information. Edges together with some adjacent grey/colour values are extracted and encoded using a classical edge detector, binary compression standards such as *JBIG* and state-of-the-art encoders such as *PAQ*. During decoding, information not covered by these encoded data is reconstructed by solving the Laplace equation, i.e. by inpainting with the steady state of a homogeneous diffusion process. The result is a simple codec that is able to encode and decode in real time. The thesis shows that for cartoon-like images this codec outperforms the successful *JPEG* standard, and even its more advanced successor *JPEG2000*.

The second part of this thesis does not aim for a fully developed compression codec. In contrast, it tries to answer the more fundamental question how to find the optimal spatial and tonal inpainting data such that an image can be reconstructed with minimised error. For this purpose besides homogeneous diffusion also inpainting with the biharmonic operator and edge-enhancing diffusion (EED), an anisotropic diffusion process, is analysed. To optimise the spatial distribution of the inpainting data, two algorithms are proposed: a probabilistic data sparsification followed by a nonlocal pixel exchange. Afterwards, the grey values in the selected pixels are optimised. In the case of linear inpainting operators an exact optimisation is possible based on a least squares approach. In the case of EED the solution of a nonlinear least squares problem is approximated. The resulting method allows almost perfect reconstructions with only 4% of all pixels even with homogeneous diffusion inpainting. With more soph-

isticated PDEs such as anisotropic diffusion, and with a simple conceptual encoding, one can already beat the quality of *JPEG2000*, even for natural images.

Finally, this thesis shows how optimised data can be integrated into the edge-based codec. While the application of the tonal optimisation is straightforward, the integration of optimal spatial data requires a few algorithmic adaptations. By maintaining the edges, but at the same time adding a few spatially optimised points, especially low contrast or blurry edges can be represented well. As a result, cartoon-like images that contain smooth variations can be compressed even more efficiently. Also the application of this codec to non-cartoon-like images delivers results of good quality, even though it is just based on homogeneous diffusion inpainting.

---

## Zusammenfassung

Die vorliegende Arbeit untersucht die Bildkompression mit partiellen Differentialgleichungen (PDEs), basierend auf Kanten und optimalen Daten.

Sie stellt zunächst ein verlustbehaftetes Kompressionsverfahren für cartoonartige Bilder vor, welches auf Kanteninformationen basiert. Dabei werden Kanten zusammen mit einigen benachbarten Grau/Farbwerten unter Verwendung eines klassischen Kantendetektors extrahiert und anschließend mittels binärer Kompressionsstandards wie *JBIG* und modernen Kodierern wie *PAQ* kodiert. Während der Dekodierung, werden Informationen, die nicht durch die kodierten Daten abgedeckt werden, als Lösung der Laplace-Gleichung rekonstruiert, also durch Inpainting mit dem stationären Zustand eines homogenen Diffusionsprozesses. Das Ergebnis ist eine einfacher Codec, der in der Lage ist, in Echtzeit zu enkodieren und dekodieren. Die vorliegende Arbeit zeigt, dass der Codec für cartoonartige Bilder den erfolgreichen *JPEG*-Standard und auch seinen weiterentwickelten Nachfolger *JPEG2000* übertrifft.

Der zweite Teil dieser Arbeit zielt nicht auf einen vollentwickelten Kompressionscodec ab. Im Gegensatz dazu wird versucht, die grundsätzlichere Frage zu beantworten, wie optimale räumliche und tonale Inpaintingdaten gefunden werden können, so dass ein Bild mit minimiertem Fehler rekonstruiert werden kann. Zu diesem Zweck werden neben homogener Diffusion auch Inpainting mittels biharmonischem Operator und kantenverstärkender Diffusion (EED) – ein anisotropes Diffusionsverfahren – analysiert. Zur Optimierung der räumlichen Verteilung der Inpainting Daten werden zwei Algorithmen vorgeschlagen: Eine probabilistische Ausdünnung gefolgt von einem nichtlokalen Pixelaustausch. Danach werden die Grauwerte in den ausgewählten Pixeln optimiert. Im Falle der linearen Inpaint-

tingoperatoren ist eine exakte Optimierung auf Grundlage der Methode der kleinsten Quadrate möglich. Im Falle von EED wird die Lösung eines nichtlinearen Kleinste-Quadrate-Problems angenähert. Das resultierende Verfahren ermöglicht nahezu perfekte Rekonstruktionen mit nur 4% aller Pixel, selbst mit homogenem Diffusionsinpainting. Mit anspruchsvolleren PDEs wie anisotroper Diffusion, und mittels einer einfachen konzeptionellen Kodierung, kann bereits die Qualität von *JPEG2000* sogar für natürliche Bilder übertroffen werden.

Schließlich zeigt diese Arbeit, wie optimierte Daten in den kantenbasierten Codec integriert werden können. Während die Anwendung der tonalen Optimierung einfach ist, erfordert die Integration der optimalen räumlichen Daten einige algorithmische Anpassungen. Indem die Kanten beibehalten werden, aber zugleich einige wenige räumlich optimierte Punkte hinzugefügt werden, können vor allem Kanten mit geringem Kontrast oder einer Unschärfe gut dargestellt werden. Somit können cartoonartige Bilder, die weiche Variationen enthalten noch effizienter komprimiert werden. Auch die Anwendung dieses Codecs auf nicht-cartoonartige Bilder liefert Resultate von guter Qualität, obwohl der Codec auf homogenem Diffusionsinpainting basiert.

---

## Acknowledgements

I would like to thank Prof. Dr. Joachim Weickert for giving me the opportunity of doing a PhD in his group, providing me with a position in research and teaching, and supervising my dissertation. I also would like to thank Prof. Dr. Søren Forchhammer for welcoming me for a research stay in his group at the Technical University of Denmark, the fruitful discussions, and for agreeing to become an external reviewer of my thesis. Furthermore, I am thankful to the Saarbrücken Graduate School of Computer Science for supporting and funding my work with a graduation scholarship towards the end of my PhD.

My special thanks goes to Pascal Gwosdek, who provided me with the Latex-template of this thesis, constantly advised and motivated me, and did not hesitate to spend his nights proofreading my thesis. Moreover, I want to thank my colleagues Martin Ebert, Georgios Dilgerakis and Janos Knobloch, who also participated in proofreading my thesis.

In addition, I want to express my gratitude to all the current and former members of the MIA group, especially Sven Grewenig, Marcus Hargarter, Oliver Demetz, Simon Setzer, Sebastian Hoffmann, Nico Persch, Christopher Schroers, and Christian Schmaltz. It was a pleasure to work with them day for day, feeling the team spirit, and enjoying their company. It goes without saying, how much I have enjoyed and benefited from discussing research ideas and work with them over countless coffee breaks. Not to mention, all the group activities we have done together, like playing soccer, climbing and table football. Those moments I will always remember and treasure.

Furthermore, I would like to thank Ellen Wintringer and Marcus Hargarter for greatly simplifying my daily work, and for always being ready to

help at any point in time. I would like to thank Andrés Bruhn for having an open ear during the many discussions of various research topics; Henning Zimmer, Levi Valgaerts, Gabriela Ghimpeteanu, Marcus Hargarter and Sebastian Hoffmann for the cherishable times sharing a common office;

Finally, I dedicate my deepest gratitude to my family and friends for their unconditional support and motivation. Special thanks goes to my friends Fabienne Eigner, Pascal Gwosdek and Martin Grochulla. I am particularly grateful to my parents Stefan and Margit Mainberger, my sister Anna Ulrich, and my brother Thomas Mainberger. They have always been there for me when I needed them. I wish to thank my wife Dina Mahmoud for pushing me not to give up, sharing the good moments with me, and helping me get through the difficult ones.



# Chapter 1

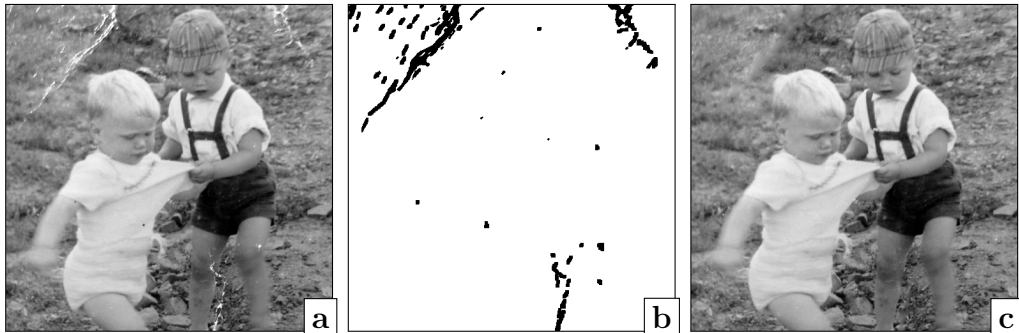
---

## Introduction

### 1.1 Motivation

We are living in an age in which mobile phones have become digital cameras, able to take high resolution pictures and able to share them almost instantly via the Internet, in which we can transfer the screen content of our laptops, notepads and mobile devices in realtime to our TVs, in which social networks are flooded by selfies, cartoons and videos of any kind and size, in which people shoot the same scene again and again, just to make sure that one of the pictures would be flawless, and in which 3D movies have found their ways into our homes. In this age which becomes more and more characterised by digital images being omnipresent in our daily life, the need for efficient ways to compress digital image information is as strong as never before.

Thereby, lossy image compression methods are of particular interest. They allow much higher compression rates than lossless ones, but at the same time supply sufficient quality by means of our human visual system. Until today, the most prominent methods in the area of lossless image compression rely on basis transforms such as the over 20 years old *JPEG* [PM92] standard, which uses a discrete cosine transform (DCT). Its more advanced successor *JPEG 2000* [TM02] was created in 2000 and intended to replace *JPEG*, but is still not widespread. Its transform is based on biorthogonal wavelets.



**Figure 1.1:** (a) Old picture, which contains defects by being crumpled. (b) Inpainting mask: Sound data is coloured white while missing information is marked by black. (c) Restored image by homogeneous diffusion inpainting.

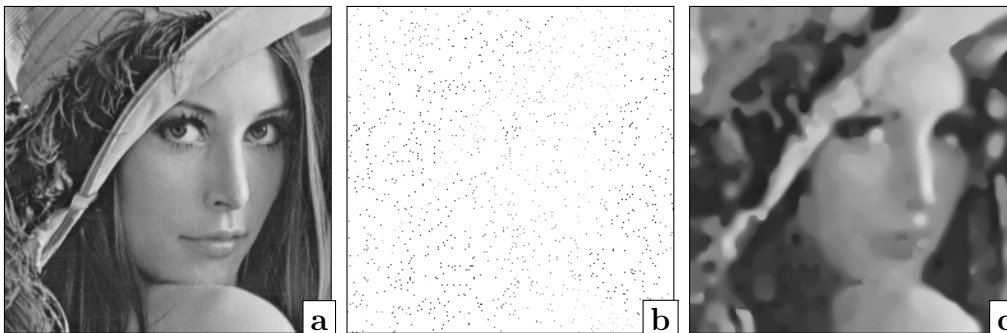
### 1.1.1 PDE-based Image Compression

As an alternative, algorithms based on *partial differential equations (PDEs)* have been gaining more and more attention in research in the last few years. In contrast to the transform-based methods, they are directly applied in the spatial image domain and thus more intuitive to understand. The underlying concept is quite simple: During encoding, a specific amount of pixels is selected and stored. For decoding, the missing information is then reconstructed by means of PDE-based inpainting [MM98a, BSCB00a, CS01a, BGS05, TD05, BM07].

In its classical application image inpainting is used for the reconstruction of lost or deteriorated image information. Thereby, a few missing parts are filled in on the basis of the given sound image data. For instance, one can restore images that contain scratches and defects as it is shown in Figure 1.1.

If we turn the amount of sound data to the other extreme so that the missing data is dominating, we reach the field of research of data interpolation. In this context PDE-based inpainting has for example been applied for upsampling digital images [RM07, WW06]. The potential of PDE-based inpainting techniques for scattered data interpolation (see Figure 1.2) has been shown in [GWW<sup>+</sup>05] and [GWW<sup>+</sup>08].

If we want to exploit the capabilities of PDE-based inpainting for image compression, there is an evident trade-off between the amount of data which is stored and the achievable reconstruction quality. This leads to two important differences to pure inpainting research: On the one hand the amount of data available for reconstruction is usually highly restricted. In this sense PDE-based image compression is closer to scattered data interpolation. On the other hand, one has the freedom to choose the data which



**Figure 1.2:** Example for scattered data interpolation with EED-inpainting as presented in [GWW<sup>+</sup>05]. **(a)** Cropped version ( $256 \times 256$ ) of the famous test image *lena*. **(b)** Scattered data: two percent of all pixels of (a), chosen randomly. Missing data is given as white pixels. **(c)** Restored image by EED-inpainting (parameters  $\lambda = 0.1$ ,  $\sigma = 1$ ).

will be available for reconstruction.

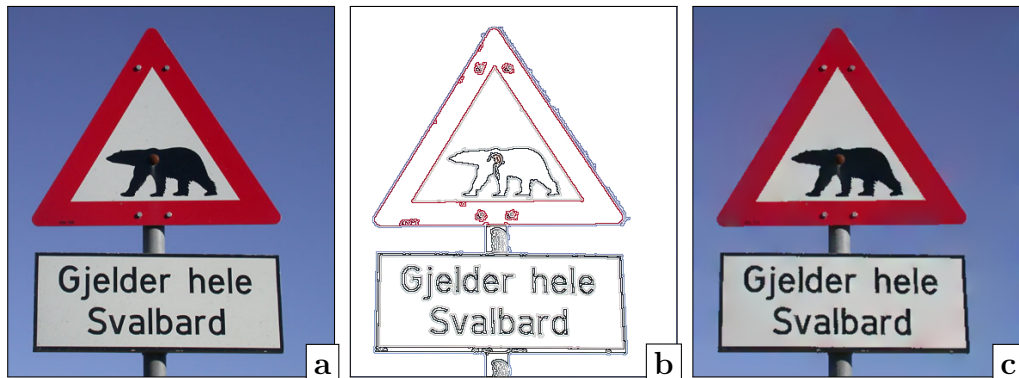
In general, compression codecs based on PDE-based inpainting are characterised by the following degrees of freedom to influence compression rate and reconstruction quality:

**The inpainting mask** is a binary image that is 1 at locations at which grey or colour information is available for inpainting during decoding, and 0 otherwise. Therefore, it determines where and also how much information is stored, which means it encodes spatial data. Actually, even non-binary inpainting masks are conceivable [HSW13]. However, in this thesis we stick to the classical approach using a binary mask.

**The Tonal Data** corresponds to the grey or colour values stored for each location defined by the inpainting mask. In classical inpainting this data is prescribed by the grey values of the original image. In image compression, they can be set to arbitrary values.

**The Differential Operator** determines how the given data is inpainted. Whereas homogeneous diffusion [Iij59] inpainting for example spreads the information evenly in all directions, more sophisticated operators that base on anisotropic diffusion [Wei96e] can even allow to reconstruct edges.

In addition the selected spatial and tonal data is most often further compressed by lossy and loss-less compression techniques, as for example entropy coding.



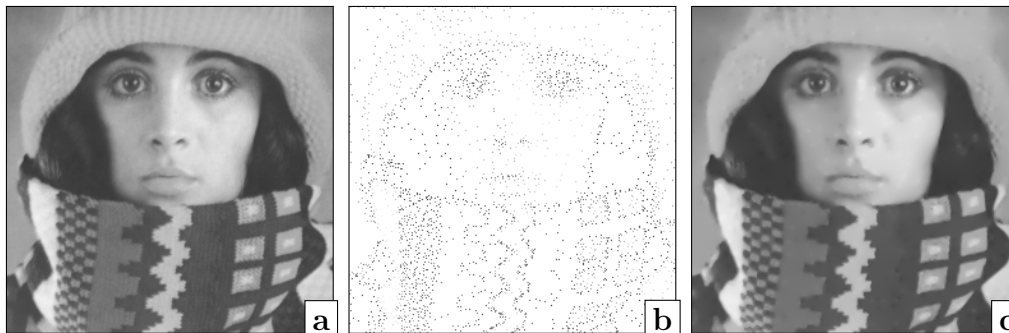
**Figure 1.3:** PDE-based image compression based on edges: (a) Test image *svalbard* ( $380 \times 431$ ) (b) Decoded data before inpainting; Missing data is given as white pixels. (c) Decoded image after compression with edge-based codec; PSNR: 30.14 dB; compression ratio 145:1.

### 1.1.2 Objective

In the first part of this thesis we present a PDE-based compression codec that is based upon the work of [MW09] and [MBWF11]. Our codec allows high quality reconstructions for cartoon-like images at high compression rates and even outperforms the quality of *JPEG2000*. In this codec the inpainting mask is described by edges and the tonal data is chosen from the adjacent grey/colour values. Homogeneous diffusion inpainting serves as differential operator. The resulting codec allows us to encode and decode images in real-time. An example is given in Figure 1.3.

The second part of the thesis extends the work presented in [MHW<sup>+</sup>12] (see also [Tan10, How10]). Whereas in the first part of the thesis, we have based the data selection on *semantically meaningful* data, namely edges, the goal here is the selection of the *optimal* spatial and tonal data by means of reconstruction quality. We show that astonishing results can be achieved with only 4% of all pixels (see Figure 1.4). Note that the second part of the thesis is more of a conceptual nature without an interest in a fully optimised and directly applicable compression codec. Instead we evaluate the theoretical limits and the potential of optimal data selection for PDE-based image compression. The insights obtained in this section may pave the way for future developments of novel efficient compression codecs.

In the last part of the thesis we use the gained knowledge to show how the edge-based codec data from the first part of the thesis can be improved by integrating optimal data.



**Figure 1.4:** PDE-based image reconstruction with optimal data: (a) Test image *trui* ( $256 \times 256$ ). (b) Decoded data before inpainting; Missing data is given as white pixels. Corresponds 4% of all pixels. Tonal and spatial data have been optimised. (c) Reconstruction with EED-inpainting.

### 1.1.3 Related Work

Let us now discuss some related work with respect to PDE-based image compression. We will not go into detail for the specific areas of edge-based image compression or choosing the optimal data for inpainting, but postpone this to the later chapters.

The simple but not less appealing idea to exploit the capabilities of PDE-based inpainting for lossy image compression was introduced first by Galić et al. [GWW<sup>+</sup>05] in 2005 and extended in 2008 [GWW<sup>+</sup>08]. In their approach, the inpainting mask is obtained by an adaptive triangulation. Thus, it can be stored efficiently as binary tree structure. Tonal data is taken from the original image and requantised. For the differential operator, an anisotropic diffusion process called *edge-enhancing diffusion (EED)* [Wei96e] is used. This codec is already able to outperform *JPEG* at low and high compression rates, but cannot exceed the compression efficiency of *JPEG2000*. The codec of Schmaltz et al. [SWB09] overcomes this problem. It is using the basic ideas of Galić et al. but improves the codec substantially by several concepts. Considering the inpainting mask, the adaptive triangulation is replaced by a subdivision in rectangles. Moreover, the grey values are adjusted by a brightness rescaling before they are requantised. With respect to EED, an optimisation of the contrast parameter within the diffusion processes of encoding and decoding, and a so-called interpolation swapping is introduced. Besides those changes Schmaltz et al. apply a better entropy coder to compress the tonal data. In [SPM<sup>+</sup>14] a revised version of this codec is presented, which improves the quality further. This paper additionally states a comprising evaluation of differential inpaint-

ing operators within the compression framework and explicitly shows that EED provides the most favourable differential operator in the context of PDE-based image compression. Furthermore, the codec is extended to 3-D data as done originally in [Pet12], and states an approach for shape-coding. In [SW12] various concepts and ideas have been combined to create a video compression codec. A progressive mode is developed in [SMMW13] and an adaptation and optimisation with respect to colour images is proposed in [PW14].

The most prominent difference of those approaches to the ones presented in this thesis is the fact that their inpainting mask is restricted to locations predetermined by the underlying tree structures. Thus, the positions of the interpolation points usually follow a pattern which makes them less optimal from a reconstruction point of view. This is the reason why those methods require more sophisticated interpolation functions, which are often based on nonlinear anisotropic diffusion processes. In contrast, our approaches even deliver astonishing results with homogeneous diffusion inpainting.

In [MSB<sup>+</sup>12] it is demonstrated how PDE-based image compression techniques can be adapted to be ideally suited for steganographic purposes. The adapted codec allows to hide large colour images within small greyscale images. Besides this, it is well-suited for uncensoring applications.

While all aforementioned codecs and the one presented in this thesis use PDE-based inpainting as their core part for compression, other approaches apply it in a preprocessing [TSYK02, KS05] or postprocessing [For96, ADF05] step.

Another work in which PDE-related approaches are used for compression is [CZ00]. The authors of this paper use total variation regularisation to modify coefficients in a wavelet decomposition in order to minimise oscillatory artefacts.

Variational approaches and PDEs can also be used for the reconstruction of surfaces. A variational taxonomy for surface reconstruction from oriented points is presented in [SSW14]. The work in [BW10] investigates the geometric diffusion equation for the interpolation of surfaces from scattered point sets. On the basis of this investigation it suggests a lossy compression method for triangulated surfaces. Examples in which PDEs are involved in the field of digital elevation maps compression are [FLA<sup>+</sup>06, SCSA04, XFC<sup>+</sup>07].

The idea to use inpainting for image compression is also exploited in [LSW<sup>+</sup>07b, RSB03, XSWL07]. In contrast to the aforementioned approaches those do not involve PDEs but integrate instead so-called structure and texture inpainting ideas in standard codecs such as *JPEG*.

There are also a number of alternative compression approaches that

are, as PDE-based compression methods, storing only a few pixels taken from the spatial domain. Those however differ in the way they reconstruct the missing data. Examples are [DNV97, DDI06, BPC09] that propose adaptive triangulations and reconstruct the missing data by interpolation within each triangle.

Furthermore, there have been methods for the reconstruction from a specific set of feature points in Gaussian scale space. Examples include [JSGA86, KLD<sup>+</sup>05] that rely on top points and [LNG03] that use a suitable set of feature points and their derivatives.

## 1.2 Outline

This thesis is organised as follows. In the next chapter we will introduce PDE-based inpainting as it is the basis of our later research on PDE-based image compression. We start with the general continuous formulation for homogeneous diffusion and embed it into the inpainting framework. The continuous formulation is then discretised giving a linear system of equations. We prove that the solution of this linear system exists and is unique. Then we give a detailed explanation how it can be computed efficiently. Furthermore, we extend the inpainting framework to the more advanced differential operators for biharmonic inpainting and inpainting with edge-enhancing diffusion (EED). We also show how to apply it to colour valued images.

In Chapter 3 we present our PDE-based compression codec that allows us to encode and decode cartoon-like images. Thereby, we first describe the encoding method, including edge detection, edge location encoding, and pixel value encoding. Afterwards we explain how the encoded image can be decoded with the help of homogeneous diffusion inpainting. In our experimental evaluation we will see that our codec can beat the quality of *JPEG* and even *JPEG2000*.

Afterwards we study in Chapter 4 how to obtain optimised spatial and tonal data regarding the reconstruction quality of PDE-based inpainting. In the first part we restrict the problem to homogeneous diffusion inpainting and start with the spatial optimisation. This includes an explanation of the method suggested by Belhachmi et al. [BBBW09]. Moreover, we develop a probabilistic sparsification method and a non-local pixel exchange that approximate the optimal positions for a fixed amount of pixels. Afterwards, we show how the results can further be improved by calculating the optimal tonal data exactly, for the set of kept pixels. We illustrate that the combination of optimised spatial and optimal tonal data allows high

quality reconstructions, even though we use homogeneous diffusion inpainting. Furthermore, in the second part of the chapter we investigate how those optimisation concepts can be used with the more advanced differential operators, which were introduced in Chapter 2. Finally we show the potential of the optimised data for PDE-based image compression. To this end we encode the obtained data using straight forward concepts as applied in Chapter 3. The results are compared with *JPEG* and *JPEG2000*.

In Chapter 5 we explain how to include optimised data into the edge-based codec presented in Chapter 3. Thereby, we reuse the methods of Chapter 4 and adapt them accordingly. We evaluate the improved codec, by comparing it to previous results.

The thesis is closed in Chapter 6. Besides an overall summary and conclusion we furthermore give an outlook on possible future work. The thesis is supplemented by a bibliography.



# Chapter 2

---

## PDE-based Inpainting

### 2.1 Motivation

In this chapter, we set the foundations for PDE-based image compression, by recapitulating PDE-based inpainting. We start in Section 2.2 by explaining the simplest differential operator in its original application: homogeneous diffusion. Then, in Section 2.3 we show how it is embedded into the inpainting framework, and state the corresponding continuous formulation for PDE-based inpainting. This formulation is, in Section 2.4, transferred to the discrete setting such that it can be applied to digital images. In this context, we also provide a proof of existence and uniqueness for the solution of the underlying inpainting scheme as it was first given in [MBWF11]. Afterwards, we proceed to show in Section 2.5 an efficient solver for computing the solution of the discrete homogeneous diffusion inpainting problem. This allows us to perform inpainting in real-time for this type of differential operators. In Section 2.6, we will additionally consider more advanced differential operators. For each of those operators, we show the continuous and discrete formulation, and present how to efficiently solve the arising systems of equations. We end the chapter in Section 2.7 with an explanation of how the methods carry over if they are applied in a multichannel setting as it is the case for colour images.

## 2.2 Homogeneous Diffusion

The term *diffusion* is derived from the Latin word *diffundere* meaning “to spread out”. It describes a mass-preserving equilibration of concentrations or temperature in a given region over time.

The simplest diffusion equation is the so-called *homogeneous heat equation* [Iij59, Wit83], which is a parabolic *partial differential equation (PDE)*. Mathematically, it can be defined as follows: Let  $u(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}$  denote a function that represents the distribution of heat at time  $t \in \mathbb{R}^+$  and location  $\mathbf{x} \in \mathbb{R}^n$ . Then, the homogeneous heat equation states relations between the partial derivatives of  $u$ :

$$\partial_t u = \Delta u . \quad (2.1)$$

Thereby,  $\partial_t u$  denotes the partial derivative w.r.t. time  $t$ , and  $\Delta u$  is the Laplace operator applied to  $u$ . In the 2-D case with  $\mathbf{x} = (x, y)^\top$ , the Laplace operator is given by  $\Delta u = \partial_{xx}u + \partial_{yy}u$ . Knowing the initial temperature distribution at time  $t = 0$  and supplementing additional boundary conditions, Equation (2.1) allows us to compute the concentration  $u$  at any location  $\mathbf{x}$  and at any time  $t > 0$ . In the following, we will refer to this process as *homogeneous diffusion*.

In image processing, we can consider images to be smooth two-dimensional functions  $f : \Omega \rightarrow \mathbb{R}$ , where  $\Omega \subset \mathbb{R}^2$  denotes a rectangular image domain. Hence, we may identify a grey value  $f(\mathbf{x})$  at location  $\mathbf{x} = (x, y)^\top$  with a specific temperature. Using  $f$  as initial state, i.e.  $u(\mathbf{x}, 0) := f(\mathbf{x})$ , we are then able to compute the evolution of the grey values over time  $t$ . To prevent information from leaving the image domain, we additionally assume reflecting boundary conditions at the boundary of our image domain  $\Omega$ , i.e.  $\partial_{\mathbf{n}}u = 0$ , where  $\mathbf{n}$  is a normal vector to the boundary.

In terms of images, homogeneous diffusion uniformly distributes the grey values more and more, while the average grey value is preserved. Eventually, a flat image is reached for  $t \rightarrow \infty$ . One can show [Wei98a] that homogeneous diffusion fulfils a maximum-minimum principle, i.e. that the values of  $u$  never leave the value range of the original image  $f$ .

Diffusion in general, and homogeneous diffusion [Iij59, Wit83] in particular, are ubiquitous in image processing and computer vision for many years. To name only a few examples, homogeneous diffusion is used in the context of smoothing and denoising [KZ96], and frequently applied as a regulariser for variational methods [HS81, WB02].

## 2.3 Homogeneous Diffusion Inpainting

If we use homogeneous diffusion in its aforementioned form, we are usually interested in a specific intermediate result, i.e., in a result  $u(\mathbf{x}, T)$  for some stopping time  $T$ . Its steady state solution, namely the average grey value, is of minor interest.

This changes as soon as we introduce additional Dirichlet boundary conditions that prescribe the grey values in specific areas  $\Omega_K$ . The effect of this modification can be explained very intuitively as follows: Let us regard the image domain  $\Omega$  as a room. Due to the reflecting boundary conditions at  $\partial\Omega$ , this room is perfectly insulated. Areas  $\Omega_K$  where Dirichlet boundaries are defined, would be represented by heaters. The heaters are all configured to individual temperatures which correspond to the prescribed grey values. During the whole process, those temperatures are kept constant. Thus, the heaters can not only have a heating effect, but also a cooling one. If we then let the temperature spread, it perfectly resembles the diffusion process described by Equation (2.1).

However, due to the heaters we now obtain a stationary temperature distribution ( $t \rightarrow \infty$ ), which is not necessarily flat anymore, but instead reflects gradual transitions between the different temperatures of the heaters. Interpreting again temperatures as grey values, this means we have a process that is able to fill in missing information in images: We use *known* grey values of the original image  $f$  to form the Dirichlet boundaries at the specified areas  $\Omega_K$ . The missing, *unknown* grey values in  $\Omega \setminus \Omega_K$  are reconstructed with diffusion. In general the process of filling in missing information in images by using given data is called *inpainting*. More specifically, in our case it is called *homogeneous diffusion inpainting* [MM98a, BSCB00a]. The set of *known* data  $\Omega_K$  is usually referred to as *inpainting data*, whereas the set of *unknown* data  $\Omega \setminus \Omega_K$  is called *inpainting domain*. Note that from a mathematical point of view, homogeneous diffusion inpainting comes down to a specific type of two-dimensional *interpolation*.

Instead of solving the parabolic PDE in Equation (2.1) for  $t \rightarrow \infty$ , we can also exploit that in this case,  $\partial_t u = 0$ . Thus, the homogeneous heat equation (2.1) can be simplified to the *Laplace equation*  $\Delta u = 0$ , and the time parameter  $t$  vanishes. As a consequence, homogeneous diffusion inpainting is now described by an elliptic PDE instead of a parabolic one. Including the boundary conditions, it reads:

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega \setminus \Omega_K, \\ u &= f && \text{on } \Omega_K, \\ \partial_{\mathbf{n}} u &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{2.2}$$

Note that for inpainting, the domain  $\Omega \setminus \Omega_K$ , for which we solve the PDE, is not necessarily rectangular anymore. However, there exists an alternative formulation which allows us to look again for a solution on the whole image domain  $\Omega$ . For this purpose, let us introduce a *confidence function*  $c(\mathbf{x})$  which determines whether a point is known or not:

$$c(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{x} \in \Omega_K, \\ 0 & \text{for } \mathbf{x} \in \Omega \setminus \Omega_K. \end{cases} \quad (2.3)$$

In the context of inpainting, this confidence function  $c(\mathbf{x})$  is often called *inpainting mask*, or simply *mask*. We can now eliminate the Dirichlet boundary conditions in Equation (2.2) by integrating them into the Laplace equation:

$$\begin{aligned} c(\mathbf{x})(u - f) - (1 - c(\mathbf{x}))\Delta u &= 0 & \text{on } \Omega, \\ \partial_{\mathbf{n}} u &= 0 & \text{on } \partial\Omega. \end{aligned} \quad (2.4)$$

By evaluating the two possible values of  $c(\mathbf{x})$  in this equation, one can easily verify that its solution is equivalent to the one given in Equation (2.2). For the remainder of this thesis we will be referring to Equation (2.4) as the *extended* formulation since the solution  $u$  is computed on the whole image domain  $\Omega$ . In contrast, we are calling Equation (2.2) the *reduced* formulation as the PDE is only solved on the subset  $\Omega \setminus \Omega_K$ .

## 2.4 Discrete Theory

As we are eventually considering digital images, we need a discrete formulation of the extended or the reduced inpainting problem given in Equation (2.2) and Equation (2.4), respectively.

To this end, let us assume we have an image with  $N$  pixels, lying on a regular grid, with grid spacing  $\mathbf{h} = (h_x, h_y)^\top$  in  $x$ - and  $y$ -direction, respectively. Moreover, we enumerate all pixels consecutively row-by-row and denote  $J = \{1, \dots, N\}$  to be the set of *all* pixel indices whereas  $K \subset J$  is the set of *known* pixel indices. The discrete version of a continuous function  $f$  is then given by a one-dimensional vector  $\mathbf{f} = (f_1, \dots, f_N)^\top = (f_i)_{i \in J}$ . Analogously, we define the solution vector  $\mathbf{u}$  and the binary *mask*  $\mathbf{c}$ , where  $c_i = 1$  if  $i \in K$  and  $c_i = 0$  otherwise. Finally, the Laplacian  $\Delta$  is discretised by means of finite differences [MG80, EBY99, MM05].

### 2.4.1 Extended Discrete Formulation

Let us start with the discrete version of the extended formulation (2.4). A straightforward discretisation yields the following system of equations:

$$c_i(u_i - f_i) - (1 - c_i)[\Delta u]_i = 0 \quad \text{for all } i \in J. \quad (2.5)$$

Thereby,  $[\Delta u]_i$  is the discretised Laplacian of  $u$  in pixel  $i$ . Taking homogeneous Neumann boundary conditions into account, it can be computed as:

$$[\Delta u]_i := \sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i)} \frac{u_j - u_i}{h_\ell^2}, \quad (2.6)$$

with  $\mathcal{N}_\ell(i)$  denoting the set of indices of pixels that are adjacent to pixel  $i$  in  $\ell$ -direction (contains up to two elements per direction). In order to rewrite the system of equations (2.5) in matrix-vector notation we define  $\mathbf{A}$  to be the symmetric, hepta-diagonal  $N \times N$  matrix with entries

$$a_{i,j} = \begin{cases} \frac{1}{h_\ell^2} & \text{if } j \in \mathcal{N}_\ell(i), \\ - \sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i)} \frac{1}{h_\ell^2} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \quad (2.7)$$

Moreover, let  $\mathbf{I}$  be the identity matrix and  $\mathbf{C} := \text{diag}(\mathbf{c})$  a diagonal matrix with the components of  $\mathbf{c}$  as diagonal entries. Then the discrete formulation of Equation (2.4) can be written as

$$\mathbf{C}(\mathbf{u} - \mathbf{f}) - (\mathbf{I} - \mathbf{C})\mathbf{A}\mathbf{u} = \mathbf{0}. \quad (2.8)$$

Thereby  $\mathbf{A}$  describes the discrete Laplace operator  $\Delta$  including the homogeneous Neumann boundary conditions. Reordering yields the linear system of equations:

$$\underbrace{(\mathbf{C} - (\mathbf{I} - \mathbf{C})\mathbf{A})}_{=: \mathbf{M}_{\text{ext}}}\mathbf{u} = \mathbf{C}\mathbf{f}. \quad (2.9)$$

The entries of the system matrix  $\mathbf{M}_{\text{ext}}$  are given by

$$m_{\text{ext}ij} = \begin{cases} -(1 - c_i)a_{ij} & \text{if } i \neq j, \\ c_i - (1 - c_i)a_{ij} & \text{if } i = j, \end{cases} \quad (2.10)$$

or equivalently when evaluating  $c_i$  by

$$m_{\text{ext}ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \text{ and } i \in K, \quad (2.11)$$

$$-a_{ij} \quad \text{if } i \in J \setminus K.$$

$u_1$	$f_2$	$u_3$
$f_4$	$u_5$	$u_6$

**Figure 2.1:** Inpainting example of size  $3 \times 2$  with two known pixels, coloured in grey.

Let us illustrate the structure of the extended system by a simple example: Figure 2.1 depicts a  $3 \times 2$  image, where  $f_2$  and  $f_4$  are known, and the grid spacing has been chosen to be  $h_x = h_y = 1$ . We then obtain the following linear system of equations:

$$\underbrace{\begin{pmatrix} 2 & -1 & 0 & | & -1 & 0 & 0 \\ 0 & 1 & 0 & | & 0 & 0 & 0 \\ 0 & -1 & 2 & | & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & | & 1 & 0 & 0 \\ 0 & -1 & 0 & | & -1 & 3 & -1 \\ 0 & 0 & -1 & | & 0 & -1 & 2 \end{pmatrix}}_{M_{\text{ext}}} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix}}_{\mathbf{u}} = \underbrace{\begin{pmatrix} 0 \\ f_2 \\ 0 \\ f_4 \\ 0 \\ 0 \end{pmatrix}}_{C\mathbf{f}}. \quad (2.12)$$

## 2.4.2 Reduced Discrete Formulation

Let us now show how the system of equations for the discretised reduced formulation (2.2) reads. Using the discrete Laplacian as given in Equation (2.6) we get for the Laplace equation  $\Delta u = 0$ :

$$\sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i)} \frac{u_j - u_i}{h_\ell^2} = 0 \quad \text{for all } i \in J \setminus K. \quad (2.13)$$

Using Dirichlet boundaries, i.e. inserting  $u_j = f_j$  for  $j \in K$ , we can rewrite these equations such that only the unknowns remain on the left hand side:

$$-\sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i) \cap J \setminus K} \frac{u_j - u_i}{h_\ell^2} = \sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i) \cap K} \frac{f_j}{h_\ell^2} \quad \text{for all } i \in J \setminus K. \quad (2.14)$$

We can also use the matrix entries of  $\mathbf{A}$  to express the Equations (2.13) and (2.14):

$$\begin{aligned} \sum_{j \in J} a_{ij} u_j &= 0 && \text{for all } i \in J \setminus K \\ \stackrel{u_j = f_j}{\text{for } j \in K} \Leftrightarrow -\sum_{j \in J \setminus K} a_{ij} u_j &= \sum_{j \in K} a_{ij} f_j && \text{for all } i \in J \setminus K. \end{aligned} \quad (2.15)$$

However, as the reduced formulation is only solved for a subset of all pixels, we cannot use  $\mathbf{A}$  to express the discrete equation system in matrix-vector notation, as we did in the extended case. Only by introducing a new reduced system matrix  $\mathbf{M}_{\text{red}}$  and vectors  $\mathbf{u}_{\text{red}}$  and  $\mathbf{b}_{\text{red}}$ , the equation system (2.15) can be written in matrix-vector notation:

$$\mathbf{M}_{\text{red}}\mathbf{u}_{\text{red}} = \mathbf{b}_{\text{red}}. \quad (2.16)$$

By  $\mathbf{u}_{\text{red}}$ , we denote the vector which contains only those  $u_i$  for which no Dirichlet data is given, i.e. with  $i \in J \setminus K$ . The right-hand-side  $\mathbf{b}_{\text{red}}$  contains for each unknown pixel the sum over the known neighbours. The system matrix  $\mathbf{M}_{\text{red}}$  is a reduced version of  $\mathbf{A}$  where every  $i$ -th row and column with  $i \in K$  has been deleted ( $i$  is the index of a know pixel). Thus,  $\mathbf{M}_{\text{red}}$  is of size  $(J \setminus K) \times (J \setminus K)$ , and since  $\mathbf{A}$  is symmetric,  $\mathbf{M}_{\text{red}}$  is symmetric as well. Note that in contrast,  $\mathbf{M}_{\text{ext}}$  is a non-symmetric,  $J \times J$  matrix.

As for the reduced formulation, let us illustrate the structure of this system by the example given in Figure 2.1:

$$\underbrace{\begin{pmatrix} 2 & 0 & | & 0 & 0 \\ 0 & 2 & | & 0 & -1 \\ \hline 0 & 0 & | & 3 & -1 \\ 0 & -1 & | & -1 & 2 \end{pmatrix}}_{\mathbf{M}_{\text{red}}} \underbrace{\begin{pmatrix} u_1 \\ u_3 \\ u_5 \\ u_6 \end{pmatrix}}_{\mathbf{u}_{\text{red}}} = \underbrace{\begin{pmatrix} f_2 + f_4 \\ f_2 \\ f_2 + f_4 \\ 0 \end{pmatrix}}_{\mathbf{b}_{\text{red}}}. \quad (2.17)$$

A comparison of the two system matrices  $\mathbf{M}_{\text{ext}}$  (see Equation (2.12)) and  $\mathbf{M}_{\text{red}}$  shows us the different treatment of known pixels: While they are “recomputed” as a part of the solution in the extended system, they have been eliminated from the solution vector completely in the reduced system. Instead, the corresponding grey values  $f_i$  are shifted from the right-hand-side of the known pixels to the right-hand-side of their neighbours.

Despite of these differences, both equation systems have two things in common: (i) They have the same solution in the unknown pixels. Thus, proofs for existence, uniqueness and a maximum-minimum principle will carry over. (ii) For a given grid spacing  $\mathbf{h}$ , they are fully described by the binary mask  $\mathbf{c} = (c_1, \dots, c_N)^\top$  and the values of  $\mathbf{f} = (f_1, \dots, f_N)^\top$ . This is important for Section 2.5, where we will consider the systems at different scales.

### 2.4.3 Discrete Well-Posedness

Now that we have established discrete formulations of our reduced and extended interpolation problems, let us show that these problems have a

unique solution that remains within the convex hull of the specified pixel data:

**Theorem 1 (Discrete Well-Posedness)**

*Let  $K$  be nonempty. Then the linear systems  $\mathbf{M}_{\text{ext}}\mathbf{u} = \mathbf{C}\mathbf{f}$  and  $\mathbf{M}_{\text{red}}\mathbf{u}_{\text{red}} = \mathbf{b}_{\text{red}}$  have a unique solution. In the unknown pixels  $i \in J \setminus K$  they satisfy the maximum-minimum principle*

$$\min_{j \in K} f_j \leq u_i \leq \max_{j \in K} f_j . \quad (2.18)$$

**Proof.**

It is sufficient to prove existence and uniqueness for the reduced problem  $\mathbf{M}_{\text{red}}\mathbf{u}_{\text{red}} = \mathbf{b}_{\text{red}}$ , since both formulations are equivalent. Thus, let us show that the system matrix  $\mathbf{M}_{\text{red}}$  is invertible. Since  $\mathbf{M}_{\text{red}}$  is symmetric, its eigenvalues are real. Moreover, by inspecting the Gerschgorin disks of  $\mathbf{M}_{\text{red}}$ , it follows that all eigenvalues are nonnegative. However, we have to exclude that 0, which can lie on the boundary of some Gerschgorin disks, is an eigenvalue. To this end, we apply a result by Feingold and Varga [FV62, Theorem 3]: If  $\mathbf{A}$  is block irreducible and  $\lambda$  is an eigenvalue of  $\mathbf{A}$  that lies on the boundary of the union of all Gerschgorin disks, then it must lie in all Gerschgorin disks. It is easy to verify that our matrix  $\mathbf{M}_{\text{red}}$  is block irreducible. Let us now consider some pixel  $i \in J \setminus K$  that has at least one pixel  $j \in K$  in its 4-neighbourhood. Then, its Gerschgorin disk  $G_i$  does not contain 0. It follows that 0 cannot be an eigenvalue of  $\mathbf{M}_{\text{red}}$  and the inverse of  $\mathbf{M}_{\text{red}}$  exists.

To prove the maximum-minimum principle, it is more convenient to consider the extended discrete model (2.9). Since  $\mathbf{M}_{\text{red}}$  is invertible, we know from the equivalence of both models that the inverse of  $\mathbf{M}_{\text{ext}}$  also exists.

First, we show that the inverse of  $\mathbf{M}_{\text{ext}}$  is nonnegative. To this end, we note that  $\mathbf{M}_{\text{ext}}$  has nonpositive off-diagonal entries, positive diagonal entries, nonnegative row sums, and at least one positive row sum. Hence,  $\mathbf{M}_{\text{ext}}$  is an M-matrix. It is well-known that the inverse of a nonsingular M-matrix is a nonnegative matrix.

Secondly we prove that each inpainted value  $u_i \in J \setminus K$  can be written as a convex combination of the specified grey values  $\{f_j \mid j \in K\}$ . Let us consider the vector of ones, i.e.  $\mathbf{e} \in \mathbb{R}^N$  with  $e_i = 1$  for all  $i \in J$ . As the



row-sum of  $\mathbf{A}$  is 0, we have

$$\mathbf{M}_{\text{ext}}\mathbf{e} = (\mathbf{C} - (\mathbf{I} - \mathbf{C})\mathbf{A})\mathbf{e} \quad (2.19)$$

$$= \mathbf{C}\mathbf{e} - (\mathbf{I} - \mathbf{C})\mathbf{A}\mathbf{e} \quad (2.20)$$

$$= \mathbf{c} - (\mathbf{I} - \mathbf{C})\mathbf{0} \quad (2.21)$$

$$= \mathbf{c}. \quad (2.22)$$

Hence, as the inverse of  $\mathbf{M}_{\text{ext}}$  exists, it holds that

$$\mathbf{e} = \mathbf{M}_{\text{ext}}^{-1}\mathbf{c}, \quad (2.23)$$

and therefore with  $\mathbf{B} := \mathbf{M}_{\text{ext}}^{-1}$

$$\sum_{j \in J} b_{i,j}c_j = \sum_{j \in K} b_{i,j} = 1 \quad \text{for all } i \in J. \quad (2.24)$$

The inpainting solution  $\mathbf{u}$  is given by

$$\mathbf{u} = \mathbf{M}_{\text{ext}}^{-1}\mathbf{C}\mathbf{f} = \mathbf{B}\mathbf{C}\mathbf{f}, \quad (2.25)$$

or considering the single vector entries, by

$$u_i = \sum_{j \in J} b_{i,j}c_j f_j = \sum_{j \in K} b_{i,j}f_j \quad \text{for all } i \in J. \quad (2.26)$$

Consequently, by Equation (2.24) and as  $\mathbf{B}$  is nonnegative, we know that  $u_i$  is in the convex hull of  $\{f_j \mid j \in K\}$ . Thus, our maximum-minimum principle is satisfied. This concludes the proof.

## 2.5 Efficient Numerical Solvers

After we have shown existence and uniqueness as well as the maximum-minimum principle for our discrete solution, let us in this section discuss how it can be efficiently computed. There are various possibilities how to do this. Among the most efficient solvers are those that use or are derived from the so-called *fast explicit diffusion (FED)* [GWB10, GWSB13, Gre13]. Their main advantage is that they are simple to implement and especially well suited for modern parallel architectures such as GPUs [GZG<sup>+</sup>10, Gwo12]. For sequential architectures like CPUs, an alternative is the so-called full multigrid method – a hierarchical iterative technique which belongs to the fastest methods for solving the Laplace equation [Bra77, Hac85]. In this

thesis we will use the latter one since it provides the best average performance across different desktop architectures, independent of the availability of programmable graphics hardware.

Let us from now on consider the extended formulation (2.9) only, as it is more convenient to work with. To simplify the notation, we rename the system matrix  $\mathbf{M} := \mathbf{M}_{\text{ext}}$ . Moreover, as the algorithm will compute solutions on different grid levels, we add to all the vectors and matrices a superscript that denotes the corresponding grid spacing  $\mathbf{h}$ .

The development of the full multigrid method is then done in four steps. First, we select a simple non-hierarchical solver that forms the basis of our multigrid implementation. Then, we show how this solver can be embedded in a two-grid cycle that performs useful correction steps at a coarser resolution. Afterwards, we focus on advanced multigrid strategies that extend this hierarchical concept to more than two grid levels. Finally, we discuss how the transfer between the different grids is accomplished.

### 2.5.1 Basic Solver

A common solver in the context of linear systems, as given by the extended formulation (2.9), is the classical Gauss-Seidel method [You71, Saa03]. The corresponding iteration step can be derived as

$$u_i^{\mathbf{h},k+1} = \frac{1}{m_{ii}^{\mathbf{h}}} \left( c_i^{\mathbf{h}} f_i^{\mathbf{h}} - \sum_{\substack{j \in J \\ j < i}} m_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k+1} - \sum_{\substack{j \in J \\ j > i}} m_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k} \right). \quad (2.27)$$

Inserting the values for  $m_{ij}^{\mathbf{h}}$  (see Equation (2.10)) yields

$$u_i^{\mathbf{h},k+1} = \frac{c_i^{\mathbf{h}} f_i^{\mathbf{h}} - \sum_{\substack{j \in J \\ j < i}} -(1 - c_i) a_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k+1} - \sum_{\substack{j \in J \\ j > i}} -(1 - c_i) a_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k}}{c_i - (1 - c_i) a_{ii}^{\mathbf{h}}} \quad (2.28)$$

$$= \frac{c_i^{\mathbf{h}} f_i^{\mathbf{h}} + (1 - c_i) \left( \sum_{\substack{j \in J \\ j < i}} a_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k+1} + \sum_{\substack{j \in J \\ j > i}} a_{ij}^{\mathbf{h}} u_j^{\mathbf{h},k} \right)}{c_i - (1 - c_i) a_{ii}^{\mathbf{h}}}, \quad (2.29)$$

and inserting the values of  $a_{ij}^h$  (see Equation (2.7)), we obtain

$$u_i^{\mathbf{h},k+1} = \frac{c_i^{\mathbf{h}} f_i^{\mathbf{h}} + (1 - c_i^{\mathbf{h}}) \left( \sum_{\ell \in \{x,y\}} \sum_{\substack{j \in \mathcal{N}_\ell(i) \\ j < i}} \frac{1}{h_\ell^2} u_j^{\mathbf{h},k+1} + \sum_{\ell \in \{x,y\}} \sum_{\substack{j \in \mathcal{N}_\ell(i) \\ j > i}} \frac{1}{h_\ell^2} u_j^{\mathbf{h},k} \right)}{c_i^{\mathbf{h}} - (1 - c_i^{\mathbf{h}}) \sum_{\ell \in \{x,y\}} \sum_{j \in \mathcal{N}_\ell(i)} \frac{1}{h_\ell^2}} \quad (2.30)$$

$$= \frac{c_i^{\mathbf{h}} f_i^{\mathbf{h}} + (1 - c_i^{\mathbf{h}}) \sum_{\ell \in \{x,y\}} \frac{1}{h_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} u_j^{\mathbf{h},k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} u_j^{\mathbf{h},k} \right)}{c_i^{\mathbf{h}} - (1 - c_i^{\mathbf{h}}) \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{h_\ell^2}}, \quad (2.31)$$

where  $|\mathcal{N}_\ell(i)|$  denotes the number of neighbours of pixel  $i$  in direction of the axis  $\ell$ ,  $\mathcal{N}_\ell^-(i) := \{j \mid j \in \mathcal{N}_\ell(i) \text{ and } j < i\}$  is the set of neighbouring pixels in direction of the axis  $\ell$  that have already been processed, while  $\mathcal{N}_\ell^+(i) := \{j \mid j \in \mathcal{N}_\ell(i) \text{ and } j > i\}$  stands for the pixels that yet have to be updated. If we finally evaluate  $c_i^h$ , we see that at known locations, we actually obtain the original data:

$$u_i^{\mathbf{h},k+1} = \begin{cases} f_i^{\mathbf{h}} & \text{for } c_i^{\mathbf{h}} = 1, \\ \frac{\sum_{\ell \in \{x,y\}} \frac{1}{h_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} u_j^{\mathbf{h},k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} u_j^{\mathbf{h},k} \right)}{- \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{h_\ell^2}} & \text{for } c_i^{\mathbf{h}} = 0. \end{cases} \quad (2.32)$$

### 2.5.2 Bidirectional Multigrid

Unfortunately, iterative solvers, such as the presented Gauss-Seidel method, have one decisive drawback: Due to the local coupling of neighbours in the iteration scheme, it may take thousands of iterations to spread information over large distances. As a consequence, only high frequencies of the error are reduced, while low frequencies remain almost undamped. This leads to a convergence rate that is very fast at the beginning, but which then slows down significantly after a few iterations.

In order to overcome this problem, bidirectional multigrid methods [Bra77, BHM00, Hac85, TOS01, Wes04] utilise the coarser levels where they obtain useful correction steps. How this works exactly, will be described

in detail by the following example of a two-grid cycle, which will form the basis of our implementation (cf. [MBWF11]).

**(1) Presmoothing Relaxation** First, we perform a few iterations with the Gauss-Seidel method given by Equation (2.31). By doing this, we reduce the high frequency components of the estimation error.

**(2) Coarse Grid Computation** The first step only gives us an approximation  $\tilde{\mathbf{u}}^h$  of the correct solution  $\mathbf{u}^h$ . To correct our result, it would be desirable to know the error  $\mathbf{w}^h = \mathbf{u}^h - \tilde{\mathbf{u}}^h$ . However, this error cannot be determined directly as we do not know the solution  $\mathbf{u}^h$ . Nevertheless, it is possible to compute the residual  $\mathbf{r}^h = \mathbf{C}^h \mathbf{f}^h - \mathbf{M}^h \tilde{\mathbf{u}}^h$  that is related to the error  $\mathbf{w}^h$  via the following equation:

$$\mathbf{M}^h \mathbf{w}^h = \mathbf{M}^h \mathbf{u}^h + \mathbf{M}^h \tilde{\mathbf{u}}^h = \mathbf{C}^h \mathbf{f}^h - \mathbf{M}^h \tilde{\mathbf{u}}^h = \mathbf{r}^h. \quad (2.33)$$

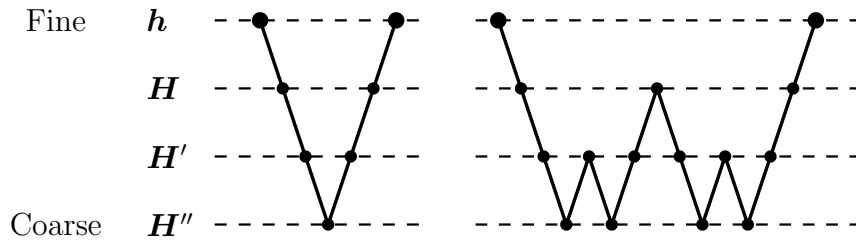
The basic idea of bidirectional multigrid methods is to transfer this so-called residual equation  $\mathbf{M}^h \mathbf{w}^h = \mathbf{r}^h$  to a coarser grid, with grid spacing  $\mathbf{H} > \mathbf{h}$ , by restricting the entries of  $\mathbf{M}^h$  and  $\mathbf{r}^h$ . Apart from the reduced computational effort on the coarse grid, this strategy offers the advantage that low frequencies on the fine grid reappear as higher ones on the coarse grid. Hence, they can be efficiently attenuated by reapplying the Gauss-Seidel solver as done in the presmoothing relaxation step. Moreover, transferring the residual equation to the coarse grid yields a system that has almost the same structure as the original one from the fine grid given by Equation (2.9):

$$\underbrace{(\mathbf{C}^H - (\mathbf{I} - \mathbf{C}^H)\mathbf{A}^H)}_{\mathbf{M}^H} \mathbf{w}^H = \mathbf{r}^H. \quad (2.34)$$

Thus the Gauss-Seidel step here looks very similar to the one from Equation (2.31):

$$w_i^{\mathbf{H},k+1} = \frac{r_i^{\mathbf{H}} + (1 - c_i^{\mathbf{H}}) \sum_{\ell \in \{x,y\}} \frac{1}{H_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} w_j^{\mathbf{H},k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} w_j^{\mathbf{H},k} \right)}{c_i^{\mathbf{H}} - (1 - c_i^{\mathbf{H}}) \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{H_\ell^2}}, \quad (2.35)$$

If the number of pixels is small enough, we use a direct solver, for example Gaussian elimination [Sch97c], to obtain the solution.



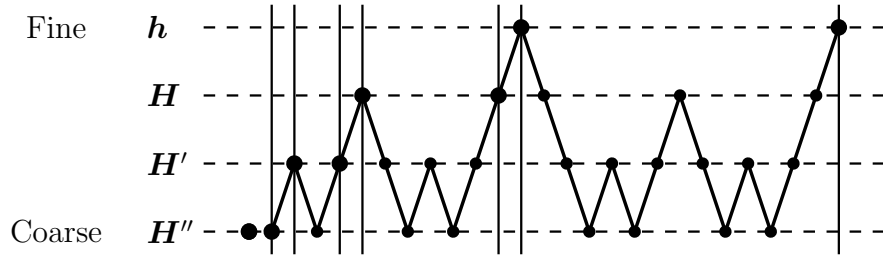
**Figure 2.2:** V-cycle (left) and W-cycle (right) for four levels with increasing resolution from  $H''$  to  $h$  (decreasing grid size).

(3) **Coarse Grid Correction** After we have solved the residual equation system on the coarse grid, we have to transfer the computed error back on the fine grid to correct our previous approximation. This correction step is given by  $\tilde{\mathbf{u}}_{\text{new}}^h = \tilde{\mathbf{u}}^h + \mathbf{w}^h$ .

(4) **Postsmoothing Relaxation** Finally, we perform again a few iterations with the original Gauss-Seidel method from Equation (2.31). This step allows us to remove high frequency errors that have been introduced by the interpolation of the coarse grid error.

### 2.5.3 Advanced Multigrid Strategies

In the previous explanation, we have only considered a two-grid cycle. However, instead of solving the residual equation system at the coarse grid directly, it is more efficient to use a third, even coarser grid that provides a correction step for the second one. By repeating this approach recursively, we obtain a so-called *V-cycle* (see Figure 2.2, left). An even better convergence rate, at the expense of slightly increased computational costs, can be reached with a so-called *W-cycle* (see Figure 2.2, right). Whenever we obtain a fine grid solution from a coarser level for the first time, we further improve this result by applying the whole procedure for a second time. Thus, we visit each coarse grid twice per corresponding fine level. Additionally, one can speed up the computation by starting with a reasonably good initialisation. To this end, we embed the *W-cycle* in a *coarse-to-fine* estimation framework: Starting from a very coarse grid, we successively refine the *original problem* (see Equation (2.9)). Solutions  $\mathbf{u}$  from coarser levels serve as initialisations of unknown pixels on finer ones. At each level, the aforementioned *W-cycle* is used to solve the resulting linear system. The combination of error correction steps and hierarchical initialisation, yields the so-called *full multigrid* method. Figure 2.3 illustrates how the differ-



**Figure 2.3:** Full multigrid scheme for four levels with increasing resolution from  $H''$  to  $h$  (decreasing grid size). At each level one  $W$ -cycle is used to solve the resulting system.

ent grids are successively traversed. In this thesis, we use a full multigrid scheme with one  $W$ -cycle per level. The choice of the number of pre- and postsmoothing iterations depends on the problem in which inpainting is applied and will be stated in the corresponding chapters.

#### 2.5.4 Transfer between Grids

The transfer between the different grids is usually achieved by a pair of so-called *restriction* and a *prolongation* operators. The restriction operator delivers the coarse grid version to the fine one. Vice versa, we obtain the fine grid version from a coarse one by applying the prolongation operator.

Considering Equation (2.34), we only need to define the restriction of the mask  $\mathbf{c}^h$  and the residual vector  $\mathbf{r}^h$  as well as the prolongation of the error  $\mathbf{e}^H$ . The discretisation of the Laplacian, i.e.  $\mathbf{A}^H$ , is redone on each level and follows directly from the corresponding grid spacings. For the coarse-to-fine strategy mentioned in Section 2.5.3, we need in addition the restriction of  $\mathbf{f}$  and the prolongation of the solution  $\mathbf{u}^H$ .

As proposed in [BWF<sup>+</sup>03], the restriction and prolongation operators can usually be realised by non-dyadic versions of area-based averaging and area-based interpolation, respectively. However, this approach leads to non-binary, blurred masks with values in the range between 0 and 1, and is thus not in accordance with the original problem. A straight-forward restriction of the original image  $\mathbf{f}$  is not possible since it is only known at specific locations. The only way is to restrict its sparse version  $\mathbf{C}\mathbf{f}$ . This in turn leads to undesired averaging effects between the known values of  $\mathbf{f}$  and the zero entries of  $\mathbf{c}$ .

As a remedy, we propose a strategy similar to normalised convolution known from scattered data interpolation [KW93]. The key idea in this con-

text is to exploit that all averaging effects become explicit in the restricted, non-binary version  $\tilde{\mathbf{c}}^H$  obtained from the binary mask  $\mathbf{c}^h$ . Thus, assuming that we have some coarse grid data  $\tilde{\mathbf{v}}^H$  given by the restriction of the sparse fine grid data  $\mathbf{v}^h := (\mathbf{C}\mathbf{f})^h$ , we define the following normalisation:

$$v_i^H := \begin{cases} \frac{\tilde{v}_i^H}{\tilde{c}_i^H} & \text{for } \tilde{c}_i^H \neq 0, \\ 0 & \text{for } \tilde{c}_i^H = 0. \end{cases} \quad (2.36)$$

By applying this normalisation in addition to the restricted mask  $\tilde{\mathbf{c}}^h$  itself, i.e.  $\tilde{\mathbf{v}}^h := \tilde{\mathbf{c}}^h$ , it becomes binary again. In contrast, since the residual vector  $\mathbf{r}$  is dense, despite of  $\mathbf{f}$  being sparse, it does not need any normalisation during restriction.

What is left is the prolongation of the error  $\mathbf{e}^H$  and the solution  $\mathbf{u}^H$  from a coarser to a finer grid. For this purpose, we apply the area-based interpolation scheme with a slight modification: We can exploit the fact that the results for  $\mathbf{u}^h$  and  $\mathbf{e}^h$  are known at mask pixels, i.e. where  $\mathbf{c}_i^h = 1$ . Thus, after prolongation we simply set back the data to the known values  $u_i^h = f_i$  and  $e_i^h = 0$  at those locations.

## 2.6 Advanced Differential Operators

Homogeneous diffusion inpainting is actually just one of the various possible alternatives that the class of so-called *PDE-based inpainting* methods offers. The type of the method is generally characterised by the differential operator that is the Laplacian  $\Delta$  in the case of homogeneous diffusion inpainting. It determines how the given data is filled into the unknown areas. In contrast to the Laplace operator, which spreads given information homogeneously, there exist more sophisticated operators, which even allow the reconstruction of edges. In this section, we will briefly discuss two of those alternatives, which will also be used later in this thesis.

Let us start by briefly rephrasing the extended inpainting formulation (2.4), where we replace the Laplace operator  $\Delta$  by a general differential operator  $L$ . PDE-based inpainting can then be more generally described by:

$$\begin{aligned} c(\mathbf{x})(u - f) - (1 - c(\mathbf{x}))Lu &= 0 & \text{on } \Omega, \\ \partial_{\mathbf{n}}u &= 0 & \text{on } \partial\Omega. \end{aligned} \quad (2.37)$$

There are various possibilities how to choose  $L$ . Intuitively, one would use more advanced diffusion operators. Besides homogeneous diffusion, there are nonlinear versions which adapt the strength of the diffusion depending

on image structures. In addition, it is possible to implement an anisotropic behaviour to steer the direction of the diffusion process [Wei98a]. Common choices, are also higher order versions of the Laplacian  $\Delta$ , i.e. using  $\Delta^p$  with some exponent  $p \in \mathbb{N}^+$ . Besides those there are even more exotic choices. Caselles et al. [CMS98] for example justified the use of the *absolute minimal Lipschitz extension (AMLE)* [Aro67]. Other PDE-based inpainting algorithms were introduced by [TS06] and [BM07]. A nice overview and evaluation of the different operators in the context of PDE-based image compression is given by [Sch12, SPM<sup>+</sup>14].

Besides homogeneous diffusion, in this thesis, we will consider the so-called *biharmonic* operator as well as *edge-enhancing (anisotropic) diffusion (EED)* [Wei96b]. Still the largest part will be concerned with with homogeneous diffusion.

Note that in contrast to homogeneous diffusion, the implementations of the inpainting methods that are based on the biharmonic operator and on nonlinear anisotropic diffusion are more complicated and considerably slower. Thus, for the examples used in this thesis, they only provide close to real-time performance. This can be remedied by efficient algorithms and hardware different to CPUs. In [KSFR07] Köstler et al. developed multigrid methods for the EED-based codec as proposed in [GWW<sup>+</sup>05]. They could show that it is possible to use these to encode videos in real-time on a *Playstation 3* device. Moreover, for EED-inpainting, efficient implementations using the FED-scheme on GPUs exist [Gwo12].

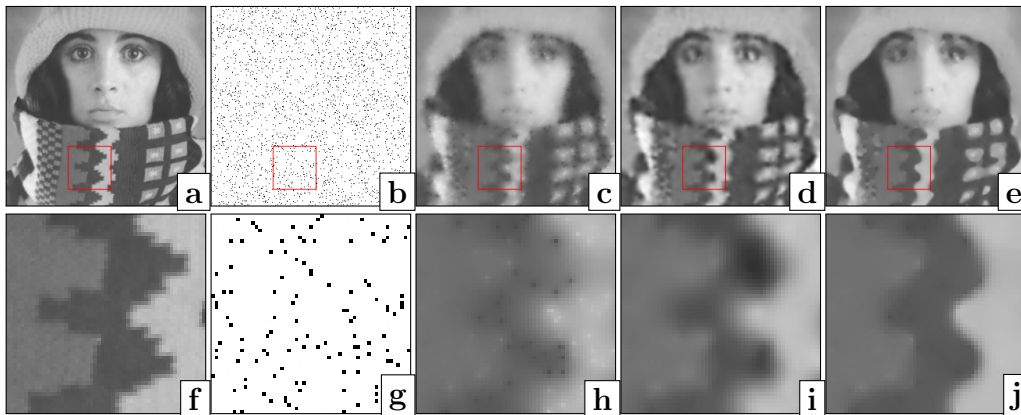
### 2.6.1 Biharmonic Operator

The *biharmonic* operator is the simplest higher order linear operator and is defined as

$$Lu := -\Delta^2 u . \quad (2.38)$$

Using it for interpolation comes down to thin plate spline interpolation [Duc76], a rotationally invariant multidimensional generalisation of cubic spline interpolation. Its corresponding Green function belongs to the family of radial basis functions [Buh03]. Compared to the Laplace operator, it yields a smoother solution  $u$  around the interpolation data. This reduces the typical singularity artefacts (see Figure 2.4) which distort the visual quality with homogeneous diffusion inpainting. On the other hand, it is prone to over- and undershoots, i.e., the values of  $u$  leave the range of the original image. This cannot happen for homogeneous diffusion inpainting which fulfils the maximum-minimum principle (cf. Section 2.4.3).





**Figure 2.4:** Reconstructions with different differential operators. (a) Original; (b) Inpainting mask, marking 4% of all pixels, randomly chosen; (c-e) Reconstructions with homogeneous diffusion (c), biharmonic operator (d) and edge-enhancing diffusion (EED) (e) when using the inpainting mask (b) and grey values as given by the original (a). **Last row:** Zoom into (a-e), region marked by red rectangle.

### Discretisation

Unfortunately, the direct discretisation of the biharmonic operator is not straightforward since the resulting expression contains fourth order (mixed) derivatives:  $-\Delta^2 u = -(u_{xxxx} + 2u_{xyyy} + u_{yyyy})$ . However, it can be rewritten as  $-\Delta^2 u = \Delta(-\Delta u)$ . The discretisation of this expression is now simple again as we can use the discrete version of the Laplacian, i.e. the matrix  $\mathbf{A}$  as defined in Equation (2.7). Hence, the discrete formulation of biharmonic inpainting reads:

$$\mathbf{C}(\mathbf{u} - \mathbf{f}) - (\mathbf{I} - \mathbf{C})\mathbf{A}(-\mathbf{A}\mathbf{u}) = \mathbf{0}. \quad (2.39)$$

If we now introduce an auxiliary vector  $\mathbf{v} := -\mathbf{A}\mathbf{u}$  the whole problem can be reformulated as

$$\mathbf{C}\mathbf{u} - (\mathbf{I} - \mathbf{C})\mathbf{A}\mathbf{v} = \mathbf{C}\mathbf{f}, \quad (2.40)$$

$$-\mathbf{A}\mathbf{u} - \mathbf{v} = \mathbf{0}, \quad (2.41)$$

which essentially means we replaced our second order problem by two first order ones. Equivalently, we can combine Equation (2.40) and (2.41) to one equation system:

$$\left( \begin{array}{c|c} \mathbf{C} & -(\mathbf{I} - \mathbf{C})\mathbf{A} \\ \hline -\mathbf{A} & -\mathbf{I} \end{array} \right) \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{C}\mathbf{f} \\ \mathbf{0} \end{pmatrix}. \quad (2.42)$$

### Numerics

This system of equations could now be solved for  $\mathbf{u}$  and  $\mathbf{v}$  with the normal Gauss-Seidel scheme. However, since there is a strong coupling of the unknowns  $\mathbf{u}$  and  $\mathbf{v}$  in this equation system, it makes sense to update in each step not only one unknown, but instead for each pixel  $i$  the two unknowns  $u_i$  and  $v_i$  at once. This can be done by solving for each pixel a  $2 \times 2$  equation system. In the literature such schemes are known as coupled point relaxation methods [Wes04]. More specifically, in combination with the Gauss-Seidel method it is called *Block-Gauss-Seidel method*. The general update step reads as follows:

$$\mathbf{B}_{ii}\mathbf{x}_i^{k+1} = \mathbf{b}_i - \sum_{\substack{j \in J \\ j < i}} \mathbf{B}_{ij}\mathbf{x}_j^{k+1} - \sum_{\substack{j \in J \\ j > i}} \mathbf{B}_{ij}\mathbf{x}_j^k \quad \text{for each block } i. \quad (2.43)$$

Thereby  $\mathbf{x}_i$  is a vector that contains  $n$  consecutive unknowns which are updated simultaneously. The original system matrix is divided into blocks  $\mathbf{B}_{ij}$  of size  $n \times n$ , i.e. each block describes a sub-matrix of the original matrix. Hence, to make the method applicable to our problem, we reorder the rows of Equation (2.42) such that the entries of the solution vector are ordered pixel-wise, i.e. it is given by  $(u_1, v_1, \dots, u_{|J|}, v_{|J|})^\top$  rather than  $(u_1, \dots, u_{|J|}, v_1, \dots, v_{|J|})^\top$ . Similar to how it was done in the previous chapter, let us illustrate the structure of the resulting equation system by the example of the  $3 \times 2$  image given by Figure 2.1, with grid spacing  $h_x = h_y = 1$ :

$$\begin{pmatrix} 0 & 2 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 2 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 3 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 2 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \\ u_5 \\ v_5 \\ u_6 \\ v_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ f_2 \\ 0 \\ 0 \\ 0 \\ f_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (2.44)$$

Note the alternating structure of the rows, which was created by reordering the solution vector. While grey coloured rows refer to Equation (2.40), white ones relate to Equation (2.41).

Applying the block Gauss-Seidel method to our equation system yields as left-hand-side of Equation (2.43)

$$\mathbf{B}_{ii}\mathbf{x}_i^{k+1} = \begin{pmatrix} c_i & -(1-c_i) \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{h_\ell^2} \\ - \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{h_\ell^2} & -1 \end{pmatrix} \begin{pmatrix} u_i^{k+1} \\ v_i^{k+1} \end{pmatrix}, \quad (2.45)$$

and as right-hand-side

$$\begin{aligned} \mathbf{b}_i - \sum_{\substack{j \in J \\ j < i}} \mathbf{B}_{ij}\mathbf{x}_j^{k+1} - \sum_{\substack{j \in J \\ j > i}} \mathbf{B}_{ij}\mathbf{x}_j^k \\ = \begin{pmatrix} c_i f_i + (1-c_i) \sum_{\ell \in \{x,y\}} \frac{1}{h_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} v_j^{k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} v_j^k \right) \\ \sum_{\ell \in \{x,y\}} \frac{1}{h_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} u_j^{k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} u_j^k \right) \end{pmatrix}. \end{aligned} \quad (2.46)$$

In order to simplify notation, we introduce the following abbreviations:

$$[\Delta_i^c] := \sum_{\ell \in \{x,y\}} \frac{|\mathcal{N}_\ell(i)|}{h_\ell^2}, \quad (2.47)$$

which is denoting the weight of the central element of the discretised Laplacian, and

$$[\Delta_i^{\mathcal{N}}(\mathbf{u})] := \sum_{\ell \in \{x,y\}} \frac{1}{h_\ell^2} \left( \sum_{j \in \mathcal{N}_\ell^-(i)} u_j^{k+1} + \sum_{j \in \mathcal{N}_\ell^+(i)} u_j^k \right). \quad (2.48)$$

which is denoting the Laplacian weighted sum over the current neighbouring grey values of  $\mathbf{u}$  (i.e. the central element is excluded). Then the equation system of the Block-Gauss-Seidel method (2.43) reads for all  $i \in J$ :

$$\begin{pmatrix} c_i & -(1-c_i) [\Delta_i^c] \\ - [\Delta_i^c] & -1 \end{pmatrix} \begin{pmatrix} u_i^{k+1} \\ v_i^{k+1} \end{pmatrix} = \begin{pmatrix} c_i f_i + (1-c_i) [\Delta_i^{\mathcal{N}}(\mathbf{v})] \\ [\Delta_i^{\mathcal{N}}(\mathbf{u})] \end{pmatrix}. \quad (2.49)$$

We can solve this  $2 \times 2$  equation system explicitly by applying for instance Cramer's rule. Sorting the obtained nominator and denominator by the mask entries, i.e. by  $c_i$  and  $(1-c_i)$ , finally gives the following update

equations:

$$u_i^{k+1} = \frac{c_i f_i - (1 - c_i) \left( [\Delta_i^c] \cdot [\Delta_i^N(\mathbf{u})] - [\Delta_i^N(\mathbf{v})] \right)}{c_i + (1 - c_i) \cdot [\Delta_i^c]^2} \quad (2.50)$$

$$v_i^{k+1} = \frac{c_i \cdot \left( [\Delta_i^c] \cdot f_i + [\Delta_i^N(\mathbf{u})] \right) + (1 - c_i) \cdot [\Delta_i^c] \cdot [\Delta_i^N(\mathbf{v})]}{-c_i - (1 - c_i) [\Delta_i^c]^2} . \quad (2.51)$$

The embedding into a full multigrid solver as done in Section 2.5 is then straight forward.

### 2.6.2 Edge-Enhancing Anisotropic Diffusion

Secondly we consider edge enhancing-diffusion [Wei96b], which is an anisotropic nonlinear diffusion operator. It can be described with the general diffusion equation:

$$Lu := \operatorname{div}(\mathbf{D} \nabla u) , \quad (2.52)$$

where,  $\mathbf{D}$  is the so-called *diffusion-tensor*, a symmetric  $2 \times 2$  matrix. As one can easily verify, we obtain homogeneous diffusion, by using the identity matrix as diffusion tensor, i.e. by setting  $\mathbf{D} := \mathbf{I}$ :

$$\operatorname{div}(\mathbf{I} \nabla u) = \operatorname{div}(\nabla u) = \Delta u . \quad (2.53)$$

In general,  $\mathbf{D}$  steers the diffusion process by its eigenvectors and eigenvalues. This explains the uniform behaviour of homogeneous diffusion inpainting, as both eigenvalues are equal. In contrast, the eigenvalues and eigenvectors of  $\mathbf{D}$  for EED are designed in dependence of  $\nabla u_\sigma$ , i.e. the gradient of the Gaussian-smoothed version  $u_\sigma$  of the image  $u$  with standard deviation  $\sigma$ . The first eigenvector  $\mathbf{v}_1$  of  $\mathbf{D}$  is chosen to be orthogonal to  $\nabla u_\sigma$ , and the corresponding eigenvalue  $\mu_1$  is fixed at 1. Since  $\nabla u_\sigma$  acts as a fuzzy edge detector, this gives full diffusion along image edges. In contrast, the second eigenvector  $\mathbf{v}_2$  is chosen to be parallel to  $\nabla u_\sigma$ , and its eigenvalue  $\mu_2$  is computed, by applying the so-called *Charbonier diffusivity* [CBAB97] to the squared gradient magnitude  $|\nabla u_\sigma|^2$ :

$$g(|\nabla u_\sigma|^2) := \frac{1}{\sqrt{1 + \frac{|\nabla u_\sigma|^2}{\lambda^2}}} . \quad (2.54)$$

Thus, across edges with high contrast the decreasing behaviour of the second eigenvalue w.r.t.  $|\nabla u_\sigma|$  reduces the diffusion. The parameter  $\lambda > 0$  allows

to steer this contrast dependence. Having defined its eigenvalue and the corresponding eigenvectors,  $\mathbf{D}$  is given by the following composition:

$$\mathbf{D} := \mu_1 \mathbf{v}_1 \mathbf{v}_1^\top + \mu_2 \mathbf{v}_2 \mathbf{v}_2^\top \quad (2.55)$$

$$= \frac{\nabla u_\sigma^\perp}{|\nabla u_\sigma|} \left( \frac{\nabla u_\sigma^\perp}{|\nabla u_\sigma|} \right)^\top + g(|\nabla u_\sigma|^2) \frac{\nabla u_\sigma}{|\nabla u_\sigma|} \left( \frac{\nabla u_\sigma}{|\nabla u_\sigma|} \right)^\top. \quad (2.56)$$

In image inpainting EED has the capability to reconstruct image edges. Due to this behaviour this operator is one of the most favourable ones especially in the context of PDE-based image compression [GWW<sup>+</sup>08, SWB09, SPM<sup>+</sup>13]. Moreover, as a diffusion process, it preserves a max-min-principle and at the same time singularities are not as frequent as they are for homogeneous inpainting.

### Discretisation

In the context of diffusion filtering, it is usually desirable that a discretisation fulfils specific scale space requirements. They lead to well-posedness and other scale space results, such as average grey level invariance, maximum-minimum principle, and convergence to a constant steady state. However, in the case of anisotropic nonlinear diffusion filtering, such as EED, it is often difficult to meet those requirements. Typical discretisations [Wei99c] of the operator (2.52) cause problems since the diagonal entries of the diffusion tensor cannot be ensured to be non-negative. As a result one of the demanded scale space requirements will be violated. There is also a so-called non-negativity discretisation [Wei98a] which gives more stable results. It fulfils the requirements if the condition number of the diffusion tensor does not exceed a value of  $3 + 2\sqrt{2} \approx 5.8284$ . However, concerning rotational invariance even this discretisation might in some scenarios not give satisfying results. In such cases the recently published discretisation framework of [WWW13] can help.

In practice, any of those discretisations could be used for our purposes and would in most cases deliver comparable results. In this thesis we will use a non-negativity discretisation as it allows sufficiently precise results. We will not go further into detail how the discretisation looks like as it is of minor interest in this thesis. Instead, we refer the reader to the respective literature mentioned above.

Even though there are different possibilities for the discretisation of the expression (2.52), its discrete equivalent is in general given by a matrix with a similar structure as the ones of the homogeneous and the biharmonic operators. However, for EED this matrix depends on the reconstruction  $\mathbf{u}$

and thus the resulting system of equations becomes nonlinear (cf. Equation (2.9)):

$$\underbrace{(\mathbf{C} - (\mathbf{I} - \mathbf{C})\mathbf{A}(\mathbf{u}))}_{=: \mathbf{M}(\mathbf{u})} \mathbf{u} = \mathbf{C}\mathbf{f} . \quad (2.57)$$

### Numerics

In order to solve it, we use a method which is also known as *time-lagged diffusivity* or *Kačanov* method [FKN73, CM99]: We replace the nonlinear system by a sequence of linear problems:

$$(\mathbf{C} - (\mathbf{I} - \mathbf{C})\mathbf{A}(\mathbf{u}^s))\mathbf{u}^{s+1} = \mathbf{C}\mathbf{f} , \quad (2.58)$$

where  $\mathbf{u}^s$  denotes the result of the  $s$ -th iteration and we set  $\mathbf{u}^0 := \mathbf{f}$  as initialisation. Those linear system of equations can then again be solved with the Gauss-Seidel method, so that we get the following update-step (cf. Equation (2.29)) for all  $i \in J$ :

$$u_i^{s+1,k+1} = \frac{c_i f_i + (1 - c_i) \left( \sum_{\substack{j \in J \\ j < i}} a_{ij}^s u_j^{s+1,k+1} + \sum_{\substack{j \in J \\ j > i}} a_{ij}^s u_j^{s+1,k} \right)}{c_i - (1 - c_i) a_{ii}^s} , \quad (2.59)$$

where we abbreviated the entries of  $\mathbf{A}(\mathbf{u}^s)$  by  $a_{ij}^s := a_{ij}(\mathbf{u}^s)$  for notational simplicity.

As for homogeneous diffusion inpainting and biharmonic inpainting we could again embed this basic solver into a full multigrid scheme (cf. Section 2.5). However, experiments have shown that the overhead necessary to perform the V/W-cycles in combination with the Kačanov- and the Gauss-Seidel iterations does not pay off in terms of runtime for the case of EED [Bru10a]. Instead, it is more convenient to replace the Gauss-Seidel method by a variant, the so-called method of *successive over-relaxation (SOR)* [Saa03] and to embed it into a pure coarse-to-fine strategy. Let us write  $\tilde{u}_i^{s+1,k+1}$  for the solution of one Gauss-Seidel step for the pixel  $i \in J$  as given in Equation (2.59). Then the SOR method replaces this solution by its pointwise extrapolation:

$$u_i^{s+1,k+1} = u_i^{s+1,k} + \omega(\tilde{u}_i^{s+1,k+1} - u_i^{s+1,k}) , \quad (2.60)$$

with a so-called relaxation parameter  $\omega \in (1, 2)$ . In our case we choose  $\omega = 1.93$ . Starting on the coarsest grid with initialisation 0 we refine the

problem step by step by using the interpolated coarse results as initialisation on the next finer level, and use the prolongation and restriction operators as explained in Section 2.5.4. For our applications it suffices to perform 20 Kačanov-steps for each level and for each Kačanov-step 5 Gauss-Seidel iterations.

## 2.7 Colour Images

So far we only considered greyscale images that were given as scalar valued functions  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ . Since we will also deal with colour images in this thesis, we have to extend the previous concepts of inpainting to vector-valued functions  $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^n$ , where  $n$  denotes the number of channels. More specifically, we will write an RGB colour image as a vector  $\mathbf{f}(\mathbf{x}) = (f_R(\mathbf{x}), f_G(\mathbf{x}), f_B(\mathbf{x}))^\top$  that contains the values for the red, green and blue channel.

Of course, the most obvious solution would be to simply apply diffusion based inpainting channel-wise. However, from the perspective of compression it is questionable whether it really pays off to store a separate mask for each channel. Often there is even a relation between the values of the different channels, such that a coupling is desirable anyway. Therefore, we will use the same mask for each channel.

It should be mentioned that there are cases in which it would be reasonable to have different masks. Using the YCbCr-colour space instead of the RGB one constitutes such an example. There, the luma component Y basically represents a greyscale version of the original colour image, which means it contains all image structures. In contrast, the chroma components Cb and Cr only measure the distance from grey in the blue-yellow and in the red-cyan directions, respectively. As a result, the chroma channels can be subsampled to a much coarser level than the luma channel. This technique is also applied in the well-known *JPEG* compression standard [PM92]. For PDE-based image compression this means, one can store much sparser masks for the chroma channels, and invest more in the Y-channel, as it is done in [PW14].

Using the same mask  $c$  for each channel, we can apply both, homogeneous diffusion inpainting as well as biharmonic inpainting, channel-wise. This is different for EED. Because it is a space-variant filter, i.e. its diffusion behaviour changes depending on the local structure, a coupling between the channels via the diffusion-tensor is desirable. This means that for EED-inpainting on RGB colour images we want to use a common diffusion tensor for all the channels.

As in the case of grey value images we would like to design it by considering the gradient of the smoothed image (see Equation (2.56)). The main question is how the gradient of a colour image is defined. Summing up the gradients of all channels is obviously not a good idea since they can cancel out, even if in all channels a strong gradient and thus an edge is present. Di Zenso [Di 86] suggests a solution to this problem: Let  $\mathbf{J}$  be the sum over the tensor products of the image gradients (in our case presmoothed) for each channel:

$$\mathbf{J} := \sum_{*\in\{R,G,B\}} \nabla u_{*\sigma} \nabla u_{*\sigma}^\top. \quad (2.61)$$

Then the gradient direction of a RGB colour image is given by the normalised eigenvector  $\mathbf{w}_1$  of  $\mathbf{J}$  corresponding to the largest eigenvalue  $\lambda_1$ . Moreover, the eigenvalue  $\lambda_1$  serves as squared gradient magnitude. Note that  $\mathbf{w}_1$  can be interpreted as the unit vector that is so to say “most parallel” to all the gradients of the different channels whereas  $\lambda_1$  describes the joint contrast in this direction. As in Equation (2.55), we can now again compose the diffusion tensor, this time for the case of RGB colour images:

$$\mathbf{D} = \mu_1 \mathbf{v}_1 \mathbf{v}_1^\top + \mu_2 \mathbf{v}_2 \mathbf{v}_2^\top := \mathbf{w}_1^\perp \mathbf{w}_1^{\perp\top} + g(\lambda_1) \cdot \mathbf{w}_1 \mathbf{w}_1^\top. \quad (2.62)$$

Note that this definition also holds for grey value images: There,  $\mathbf{J}$  simplifies to  $\nabla u_\sigma \nabla u_\sigma^\top$ . The normalised eigenvectors of  $\mathbf{J}$  are then given by

$$\mathbf{w}_1 = \frac{\nabla u_\sigma}{|\nabla u_\sigma|} \text{ and } \mathbf{w}_2 = \frac{\nabla u_\sigma^\perp}{|\nabla u_\sigma|}. \quad (2.63)$$

with associated eigenvalues  $\lambda_1 = |\nabla u_\sigma|^2$  and  $\lambda_2 = 0$ , respectively. If we insert those values in Equation (2.62), we arrive at Equation (2.56).

Now that we have defined the diffusion tensor for EED, the discretisation can be carried out for all the operators in each channel exactly as in the grey valued case.



# Chapter 3

---

## Edge-based Image Compression

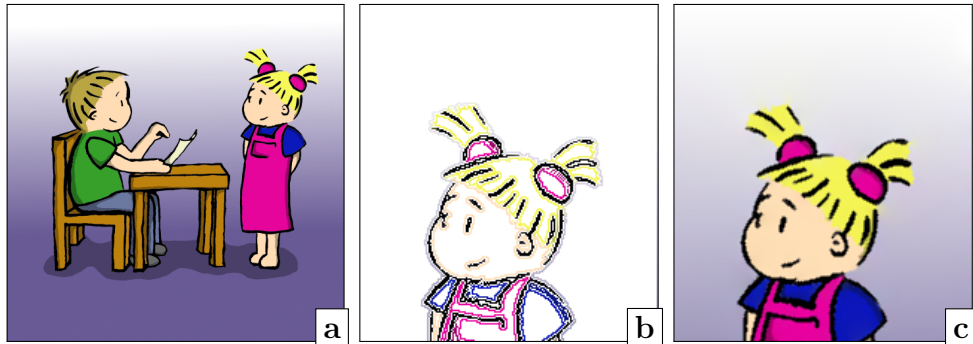
### 3.1 Motivation

The fact that we are able to understand cartoons and line-drawings indicates that edges provide perceptual meaningful data. This makes them also an interesting feature for image processing and computer vision purposes where edges are often understood as an intermediate step from a pixel-based to a semantic image representation.

Since it is more compact to describe an image by a few contours than by many pixels, edges are also of potential interest for image compression.

In general, approaches that incorporate properties of the human visual system into image compression belong to the class of so-called second-generation image coding methods [KIK85, RMB97]. They usually do not rely on basis transforms, as it is the case for *JPEG* and *JPEG2000*. Instead, they extract perceptually relevant features of the image and neglect visually insignificant data. Hence, those methods are in general also lossy.

Regarding the specific field of edge-based coding, there exist numerous theoretical and experimental papers [BL77, COL85, YP86, ZR86, Che87, Cai88, HM89, GG90, MZ92a, Dro93, SH93, AG94, AD96, DMMH96, Eld99, Mai08, WZSG09] which have shown that in general reconstructions from edge data are possible. However, it has been experienced that knowing the locations of edges alone is not sufficient to reconstruct an image. Therefore, the mentioned papers make different suggestions about additional data to be added in order to obtain “complete” reconstructions. Some suggest to incorporate gradient information, others propose to store the grey values adjacent to the edges. It is also possible to consider subsampled image

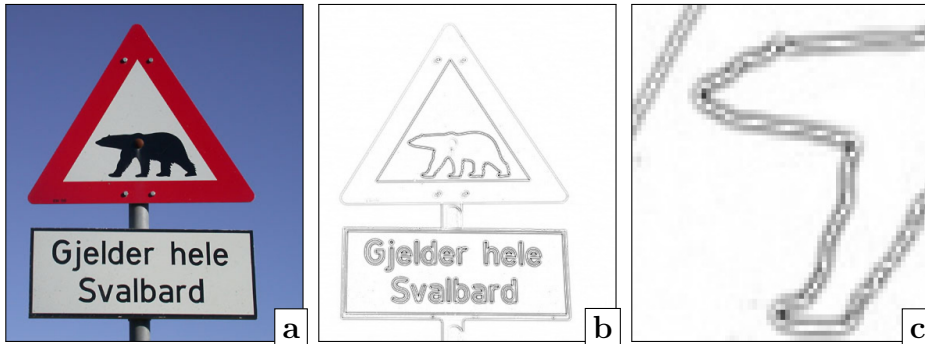


**Figure 3.1:** (a) Test image *comic* ( $512 \times 512$ , synthetic). (b) Zoom into reconstructed colour pixels adjacent to the edges of the test image *comic*. Undefined pixels are coloured white. (c) Zoom into corresponding reconstruction with homogeneous diffusion inpainting.

data that lies not directly at the edge or to include some scale information. Not all the mentioned papers used their reconstruction approaches with the goal of application to image compression. However, those who did were not able to come up with competitive results to compression standards such as *JPEG* or *JPEG2000*. Either the required data could not be encoded in a sufficiently compact way or the results turn out to be of inferior quality.

Only recently, we have presented in [MW09, MBWF11] that such approaches can clearly outperform the quality of *JPEG2000* for cartoon-like images. In this chapter, we are going to discuss this edge-based compression algorithm. The basic idea is appealingly simple. During encoding we first detect the edges of the original image. In a second step, we consecutively collect only those grey or colour values that lie in the direct neighbourhood of the edges. The obtained values are then further compressed and finally stored together with the locations of the extracted edges. During decoding, the colour values are again distributed along the edges. They now form the Dirichlet data for inpainting (see Chapter 2.3). Since edges usually split areas of different brightness or colour we end up with sufficient information to fill-in the missing areas between the edges. Figure 3.1 shows an exemplary result.

In our compression codec, we use homogeneous diffusion inpainting. We prefer this simple inpainting approach in this setting over more advanced ones such as biharmonic- or EED-inpainting for several reasons: First of all, it is one of the analytically best understood inpainting approaches. Recently, Belhachmi et al. [BBBW09] presented a continuous analysis on spatially optimal data selection for homogeneous diffusion interpolation.



**Figure 3.2:** (a) Test image *svalbard* ( $380 \times 431$ , real world). (b) Modulus of the Laplacian of (a) normalised to  $[0,255]$ . The darker the colour, the larger the modulus of the Laplacian. (c) Zoom into (b).

Their framework is based on the theory of shape optimisation and suggests to choose the interpolation data proportional to the modulus of the Laplacian of the image<sup>1</sup>. As illustrated in Figure 3.2, this corresponds for cartoon-like images to the pixels left and right of an edge contour. Hence, by choosing homogeneous diffusion for inpainting and by using the pixels adjacent to edges as inpainting data, we meet the suggestion of Belhachmi et al. [BBBW09] to a certain extent.

Secondly, homogeneous diffusion inpainting is the simplest and computationally most favourable inpainting approach based on partial differential equations (PDEs) [MM98a]. Using the full multigrid scheme as introduced in Section 2.5, our codec is not only able to encode but also to decode images in real time. This would be hardly possible with biharmonic or EED-inpainting.

Moreover, one of the reasons for the bad reputation of homogeneous diffusion for inpainting tasks is that individual points may create unpleasant singularities in the solution of the Laplace equation (cf. Figure 2.4(c) and (h)). Since we use connected contours this problem does not appear in our case.

Last but not least, in 1935 Werner already stated an interesting hypothesis [Wer35]: He claimed that a contour-based filling-in process is responsible for the human perception of surface brightness and colour. To this end, filling-in information from image edges resembles a classical finding in biological vision and thus again meets the spirit of second generation image coding.

<sup>1</sup> We will come back to this approach in the next chapter, see Section 4.2.1

As a result of our work, there have been papers with modified versions of our codec tailored to the compression of depth maps [LSJO12, GMG12]. Even though depth maps look cartoon-like at first glance, it turns out that our codec gives unsatisfactory results in this case. This is because homogeneous variations cannot be represented well by our codec. As a remedy the modified versions add further pixels values at regular grid points. Other changes are the usage of different edge detectors and alternative encoding techniques of the extracted data.

In [HMWP13], this concept was further improved. By replacing the edges by segment boundaries and performing the inpainting within each segment separately, the colour information along the edges became obsolete. Moreover, the quality has been increased by including optimisation techniques such as the ones we will introduce in the next chapter.

Closely related ideas to our approach have recently also found a lot of attention in the computer graphics domain. So-called diffusion curves [OBW<sup>+</sup>08] consist of a few smoothly varying colour values along each side of a scalable curve. Similar to our approach, the colours from each side of the curve are spread into the image. This allows artists to create resolution-independent artworks. Furthermore, representing extracted edges of an image with Bézier splines allows an automatic conversion of pixel based images into a representation with diffusion curves. However, there are also indicators that an application to image compression in its current form would likely require further improvements [OBB<sup>+</sup>13].

Finally, there is related work that also employs the idea of combining inpainting and edges for the purpose of image compression [WSWX06, SWL06, LSW<sup>+</sup>07b]. In contrast to our codec, those apply structure and texture inpainting techniques instead of PDE-based ones.

This chapter is structured as follows: Section 3.2 describes the encoding method, including edge detection, edge location encoding, and pixel value encoding. In Section 3.3 we briefly discuss the decoding. After an experimental evaluation in Section 3.4 we conclude the chapter with a short summary in Section 3.5.

## 3.2 Encoding

In this section we explain the encoding phase, which essentially consists of three steps. First of all, edges are extracted which encode the location of adjacent grey or colour values. The second step is to encode these locations efficiently. The last step addresses the encoding of the grey or colour values.

### 3.2.1 Edge Detection

Even though until today plenty of edge detectors have been developed, we will use one of the oldest and most classical edge detectors: The Marr-Hildreth edge detector [MH80]. It extracts edges as zero-crossings of the Laplacian of a Gaussian presmoothed image  $f_\sigma$ , where  $\sigma$  denotes the standard deviation of the Gaussian kernel. For an RGB colour image  $\mathbf{f} = (f_R, f_G, f_B)^\top$  we extract the zero-crossings of the sum of the Laplacians over all channels:

$$\Delta \mathbf{f} = \sum_{* \in \{R, G, B\}} \Delta f_* = \Delta \left( \sum_{* \in \{R, G, B\}} f_* \right). \quad (3.1)$$

In this form, the Marr-Hildreth edge detector is obviously very simple to implement and at the same time one of the fastest edge detectors. Moreover, it suits well the theory of Bellhachmi et al. [BBBW09], mentioned in the previous section: Evidently at the zero-crossing itself the Laplacian magnitude is zero, whereas it is larger than zero everywhere else. At sharp edges the Laplacian magnitude is largest directly to the left and the right side of the zero-crossing (see Figure 3.2).

As the Laplacian will be zero in homogeneous regions as well, and those zero-crossings obviously have no perceptual significance, we combine the Marr-Hildreth edge detector with hysteresis thresholding as suggested by Canny [Can86]. To this end we first identify *edge candidates* as zero-crossing pixels where the *edge magnitude* exceeds a given threshold  $T_1$ . Thereby the edge magnitude of a grey value image is defined to be the gradient magnitude of the original image. In the case of colour images, it is given as the length of the vector  $(|\nabla f_R|, |\nabla f_G|, |\nabla f_B|)^\top$ , i.e. as  $\sqrt{|\nabla f_R|^2 + |\nabla f_G|^2 + |\nabla f_B|^2}$ . The gradients are computed using Sobel operators. Next, all edge candidates with an edge magnitude that is larger than a threshold  $T_2 > T_1$  become seed points for relevant edges. Moreover, those pixels are already considered to be final edge pixels. Afterwards, we recursively add all edge candidates that are adjacent to final edge pixels. This allows to keep edge pixels connected as much as possible.

In general, we are of course not strictly bound to this specific edge detector. However, modern edge detectors often aim at the detection of object boundaries and are thus not necessarily well-suited for our purposes. Other simple edge detectors such as the Canny edge detector [Can86] or methods based on the Tobbogan watershed segmentation [Fai90] deliver comparable results. Still our experiments have shown that best results for cartoon-like images, can usually be obtained with a zero-crossings-based edge detector.

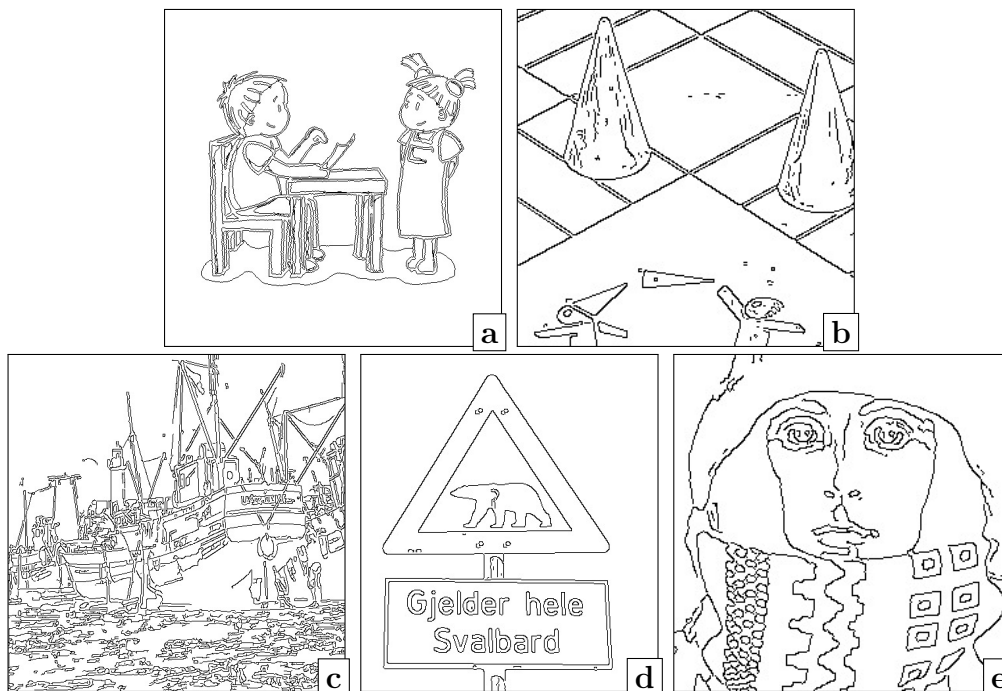
### 3.2.2 Encoding the Contour Location

As we will see, the detected edges indirectly encode the mask for the inpainting process during decoding. Hence, it is necessary to store this information in an efficient way. Since the edges of an image can be visualised as a bi-level image as depicted for example in Figure 3.3, it seems to be natural to simply store this bi-level image with a suitable method. For the compression of bi-level images several methods exist.

The so-called *JBIG* (*Joint Bi-level Image Experts Group*) standard [Joi93] has been developed as a specialised method for the compression of bi-level images, mainly with respect to telefax transmission images which mostly contain textual and line drawing information. An *edge image* – that is how we denote a bi-level image that is depicting edges from now on – actually provides quite similar data. In this thesis we use *JBIG-KIT* [Kuh95], which is a free C implementation of the *JBIG* coder. To get optimal compression results for our data, we apply the method in its non-progressive mode that also automatically excludes the so-called *deterministic prediction*. Furthermore, we disable the *typical prediction* step of *JBIG*. Usually the *JBIG* standard allows to subdivide an image into stripes and encodes these stripes separately. We prevented the routine from doing this such that the image is encoded in its entirety. Finally we set the maximum adaptive template pixel offset to 0. With these settings, the routine essentially comes down to a context-based arithmetic coder using a static template for prediction.

*JBIG2* [HKM<sup>+</sup>98, Joi99] is the successor of *JBIG* and can for example be found as a compression scheme in PDF files versions 1.4 and above. In this thesis we use the open source encoder `jbig2enc` (Version 0.27) as well as the open source decoder `jbig2dec` (Version 0.10). *JBIG2* offers a so-called symbol mode. In this mode it tries to group the (textual) data into symbols that are stored in a dictionary. This dictionary is encoded using context-dependent arithmetic coding. Since similar looking symbols are represented by a single bitmap `jbig2enc` is lossy. If the symbol mode is disabled the coder typically relies on a context-based arithmetic coding algorithm as *JBIG* does, and is lossless.

Last but not least we want to consider the computer file format *DjVu* [BHH<sup>+</sup>98]. It was designed to store scanned documents. This comprises a combination of line drawings and text, but also photographs. For pure bi-tonal images the open source DjVu-library *DjVuLibre* offers an encoder (`cjb2`) and a decoder (`ddjvu`). The method used by `cjb2` is called JB2 and is similar to the symbol mode of *JBIG2*. `cjb2` provides a lossless and a lossy mode. The lossy mode allows small changes on the input image



**Figure 3.3:** Test set of edge images: (a) *comic* ( $512 \times 512$ ), (b) *coppit* ( $256 \times 256$ ), (c) *boats* ( $512 \times 512$ ), (d) *svalbard* ( $380 \times 431$ ), (e) *trui* ( $256 \times 256$ ).

in order to improve the compression ratio. We applied `cjb2` in its lossless mode and in its highest possible lossy setting (i.e. `losslevel=200`).

Our codec allows to choose any of the presented coders by setting a flag in the file header. Theoretically, if time is not an issue one could let the algorithm try all different methods and always pick the one which gives the best result. However, if we analyse which coder offers in general the best performance on edge images, such an optimisation step seems to be a waste of time: Table 3.1 gives a comparison of the methods applied to a test set of five different edge images (see Figure 3.3). The results clearly suggest to favour *JBIG* for the encoding of our data. Even the lossy mode of DjVu cannot beat its compression rates. In addition, *JBIG* is most often the fastest of all presented methods. Thus, for the results shown in Section 3.4 we restrict ourselves to the *JBIG* coder.

### 3.2.3 Encoding the Contour Pixel Values

Besides the edge image, we also need to store the grey or colour values needed for inpainting. As mentioned in the previous sections, those pixel

**Table 3.1:** Comparison of different bi-tonal coders regarding their compression ratio and time when encoding the edge images from Figure 3.3. Coders: *JBIG*, *JBIG2* with generic coder, *JBIG2* with symbol-mode (*JBIG2S*), *DjVu* lossless and *DjVu* losslevel 200 (*DjVuL*). CPU: Intel Core 2 Duo T7500 @ 2.20Ghz. The best result in each line is in bold face letters.

coder	Edge image				
	<i>comic</i>	<i>coppit</i>	<i>boats</i>	<i>svalbard</i>	<i>trui</i>
<b>compression ratio in bpp:</b>					
<i>JBIG</i>	<b>0.099</b>	<b>0.124</b>	<b>0.256</b>	<b>0.044</b>	<b>0.173</b>
<i>JBIG2</i>	0.109	0.137	0.273	0.053	0.198
<i>JBIG2S</i>	0.113	0.158	0.304	0.057	0.217
<i>DjVu</i>	0.105	0.160	0.298	0.054	0.199
<i>DjVuL</i>	0.104	0.143	0.295	0.053	0.198
<b>encoding time in ms:</b>					
<i>JBIG</i>	<b>3</b>	<b>1</b>	<b>4</b>	<b>3</b>	<b>&lt;1</b>
<i>JBIG2</i>	4	<b>1</b>	5	5	2
<i>JBIG2S</i>	8	7	19	5	5
<i>DjVu</i>	21	16	29	13	14
<i>DjVuL</i>	28	15	45	17	23
<b>decoding time in ms:</b>					
<i>JBIG</i>	3	<1	<b>2</b>	3	1
<i>JBIG2</i>	4	1	5	<b>1</b>	<1
<i>JBIG2S</i>	<b>2</b>	3	6	4	1
<i>DjVu</i>	13	16	22	16	21
<i>DjVuL</i>	10	18	18	21	14



values are located adjacent to the extracted edges. In addition, we want to store all pixel values from the border of the image domain. Experiments have shown that this gives better results when reconstructing missing information during decoding.

### Ordering of the pixel values

The simplest order in which the pixel values could be stored would be row by row or column by column by the order of their occurrence. However, pixel values of opposite edge sides usually differ by a considerable amount. In contrast, pixel values along one side of a contour usually change only gradually. Thus, a collection of the pixel values by the order of their occurrence along the edges decreases the entropy of the retrieved pixel value stream. Entropy coders, as we will apply them later on, can benefit from this ordering and thus offer higher compression rates.

It turns out that collecting the values properly along a contour side is not as trivial as it seems at first glance. Often edges lie that close to each other that they share several neighbours. Let us now assume we have a strategy that would first collect all direct pixels of the first edge. While tracing the second edge, we would of course not like to recollect pixels values that have already been fetched for the first edge. Instead, we would only pick the remaining pixels values. As a result the values of the second edge would appear highly fragmented, which may lead to high contrast differences in the pixel value stream. Hence, the entropy would increase.

To overcome this problem it is desirable to have a method that fetches not only the direct but also adjacent pixel values in a certain neighbourhood along the edges. That means pixel values of edges that are close to each other would then be collected at once. Since those values are anyway expected to have similar brightness or colour, we prevent the entropy to increase.

Let  $\mathbf{f}$  be a grey value image, and let  $K$  be the set of mask pixel indices (cf. Chapter 3), i.e. the indices of the pixels which should be available for reconstruction when decoding. Here, they are given by those pixels that are adjacent to edges and in addition the border pixels of the image  $\mathbf{f}$ . The following algorithm, which we explain below, yields a 1-D output stream  $\mathbf{s}$  that contains the desired pixel values.

**Algorithm: Pixel tracing**

---

**Input:**

Set of mask pixel indices  $K$ , original image  $\mathbf{f}$ , distance  $d_{\text{tr}}$ .

**Initialisation:**

$K_{\text{visit}} := K$ , empty queues  $Q_1$  and  $Q_2$ .

**Compute:**

For all  $i \in K$ :

1. If  $i \in K_{\text{visit}}$ , initialise  $Q_1$  with  $i$ .
2. While  $Q_1$  is not empty do:
  - a. Get and remove a pixel index  $i$  from  $Q_1$ .
  - b. If  $i \in K_{\text{visit}}$  initialise  $Q_2$  with  $i$ , remove  $i$  from  $K_{\text{visit}}$ , append  $f_i$  to  $\mathbf{s}$  and set  $i_{\text{last}} := i$ .
  - c. While  $Q_2$  is not empty do:
    - i. Get and remove a pixel index  $i$  from  $Q_2$ .
    - ii. For each pixel  $i_{\mathcal{N}}$  in the 4-neighbourhood of  $i$  with  $i_{\mathcal{N}} \in K_{\text{visit}}$  do:
 

If spatial distance between the pixels  $i_{\mathcal{N}}$  and  $i_{\text{last}}$  is larger than  $d_{\text{tr}}$

Put  $i_{\mathcal{N}}$  on  $Q_1$

else

Put  $i_{\mathcal{N}}$  on  $Q_2$ , remove it from  $K_{\text{visit}}$ , append  $f_{i_{\mathcal{N}}}$  to  $\mathbf{s}$  and set  $i_{\text{last}} := i_{\mathcal{N}}$ .

**Output:**

Pixel value stream  $\mathbf{s}$ .

---

Note that in the following subsection, we may refer to a pixel's index when saying pixel, depending on the context. We start our algorithm by creating a set  $K_{\text{visit}}$  that initially contains all pixels of the inpainting mask. Using this set, we keep track of the pixels that still have to be visited. Thus, whenever a pixel value is added to the output stream  $\mathbf{s}$ , we remove it from the set  $K_{\text{visit}}$ . Moreover, we add it to a queue  $Q_2$  as we want to explore its 4-neighbourhood for unvisited pixels. Last but not least we temporarily save it in an auxiliary variable  $i_{\text{last}}$  so that we know at any time which pixel contributed last to the output stream.

The core of the algorithm is then essentially given by the loop starting in Step 2c. There, we retrieve pixels from  $Q_2$  until this queue is empty. In the very first run  $Q_2$  contains only one pixel, namely the one corresponding to the first and only pixel value which has been added to the output stream  $\mathbf{s}$  so far. Now we want to explore its neighbours one after the other. For any neighbour that has not been visited yet and thus is still in  $K_{\text{visit}}$  we make a decision: If its distance to the pixel whose value has been added last to  $\mathbf{s}$  (i.e.  $i_{\text{last}}$ ) is still smaller than  $d_{\text{tr}}$ , we append its value to  $\mathbf{s}$  and consequently we also remove it from  $K_{\text{visit}}$ , add it to  $Q_2$  and store it in  $i_{\text{last}}$ . However, if the distance exceeds  $d_{\text{tr}}$ , we regard the currently considered pixel to be too far away from the one whose value has been stored last. That means the differences between their values are probably high. In this case we postpone its processing by adding it to a queue  $Q_1$ . As soon as  $Q_2$  is empty, we reinitialise  $Q_2$  with the next pixel from  $Q_1$ , provided it has still not been visited in the meantime. As soon as all pixels that are 4-connected to the first one have been visited,  $Q_1$  will be empty as well. Only in this case we reinitialise  $Q_1$  with another unvisited pixel from  $K_{\text{visit}}$ . Since the whole procedure is repeated until all mask pixels have been visited, i.e.  $K_{\text{visit}}$  is empty, the algorithm is guaranteed to terminate.

In contrast to a pure depth-first search this strategy collects not only the direct pixel values along an edge. Depending on  $d_{\text{tr}}$  also reachable pixel values which lie up to a certain distance close to the edge can be found.

Note that for the sake of simplicity, the algorithm above was given for the case of a grey value image. It is obvious that in the case of colour images we simply append the pixel values of each channel to three different output streams  $\mathbf{s}_*$  with  $* \in \{R, G, B\}$ .

### Subsampling

Our tracing strategy from the previous section takes care to collect the pixel values in the order of their appearance along the edge. Apart from the fact that entropy coders can benefit from such an ordering, it also allows us to reduce the amount data in another way. Instead of storing all collected pixel values, we only keep every  $d$ -th value which means we perform a uniform subsampling of each pixel value stream. Since pixel values along an edge change only marginally, the missing pixels can most often be reconstructed reasonably well with linear interpolation during decoding. The proper reconstruction only fails when neighbouring sample points belong to distinct edges that do not share any pixels. In this case, the grey or colour values of those edges will mix and usually lead to a distorted result.

As a remedy, we consider how the pixel values along edges have been collected (see previous section). Whenever the queue  $Q_2$  is empty, we assume the previously collected pixels to belong to the same edge segment as there were no more neighbours to be explored. For each edge segment, we then perform the subsampling separately. We keep both, the start and the end point of each segment. Note that this approach does not demand to store any additional information since it fully relies on the stored edge image.

According to the sampling theorem, the quality of a reconstructed signal can be improved by presmoothing the original signal. In our method, we suggest to smooth the separated 1-D signals by a Gaussian convolution with standard deviation 1, assuming a grid size of 1. This removes small variations, and thus also improves the compression rate of the entropy coder later on. Furthermore, smoothing includes some neighbourhood information of the removed pixels into the sampled pixels.

In the case of colour images we allow that each stream  $\mathbf{s}_*$  is subsampled with a different resolution, i.e. we have different sampling distances  $d_*$  for each channel  $*$ . In the case of RGB images such a different subsampling usually does not pay off, as the importance of all three channels is equally high. In contrast, with other colour spaces such as YCbCr it is quite common to have different resolutions for the different channels. We have experienced that our codec could not benefit from such a colour transform.

We also tried different non-uniform subsampling strategies, including a tree-based approach, as well as setting the locations optimally. However, those approaches need to store additional overhead to encode the locations of the sample positions. This additional effort in storage did not pay off in terms of quality. The results could also not be improved when using higher order interpolation methods, such as cubic B-spline interpolation, for the reconstruction of missing information between sample points.

### Requantisation

Another way to reduce the data further, is a requantisation of the pixel values. Originally, our image consists of 256 different pixel values per channel. By requantising, we reduce the co-domain to  $q$  different values. One of the simplest quantisation approaches is the so-called *midtread quantisation*. This is a uniform quantisation that allows to reconstruct the minimal and maximal value of the original range.

Let  $s \in \{0, \dots, 255\}$  be an original value, and let  $a := \frac{255}{q-1}$ . Then the

quantised value is given by

$$s_{\text{quantised}} = \left\lfloor \frac{s}{a} + \frac{1}{2} \right\rfloor, \quad (3.2)$$

where  $\lfloor \cdot \rfloor$  denotes the floor function, i.e.  $\lfloor x \rfloor$  is the largest integer that does not exceed  $x$ . As a result, we obtain  $s_{\text{quantised}} \in \{0, \dots, q-1\}$ . In order to reconstruct the value, we simply compute

$$s \approx a \cdot s_{\text{quantised}}. \quad (3.3)$$

Note that this means we subdivide our original range in  $q$  intervals. All intervals have the width  $a$ , except the last and first one which have width  $\frac{a}{2}$ . After reconstruction, all original values within the first interval have been mapped to 0, whereas the values of the last interval have been mapped to 255. The original values of the other intervals have been mapped to the central value of the interval. For colour images, our method allows to requantise each channel separately, i.e. we have  $q_*$  with  $*$  denoting the  $*$ -th channel.

Instead of using a uniform quantisation method, it can pay off to use a non-uniform method as for example the Max-Lloyd quantiser [Max60, Llo82]. The goal is to adapt the width of the intervals to the probability at which the pixel values occur in the original signal: We start with uniform quantisation. Instead of mapping the values of an interval to the central value, we map it to the centre of mass within this interval. We will refer to these points as reconstruction points. Then, the interval boundaries are adapted such that they lie again exactly between two reconstruction points.

Starting from this setting, we repeat the whole procedure until the interval boundaries do not change anymore. Using the obtained reconstruction points, we achieve better reconstructions. However, for this quantiser we have to store the reconstruction points additionally to reconstruct the pixel values during decoding.

In our case Max-Lloyd quantisation did not pay off, except when using only a few quantisation levels. Thus, our codec uses uniform quantisation in channel  $*$  if  $q_* > 8$  and Max-Lloyd quantisation otherwise.

## Entropy Coding

Now that we have subsampled and requantised the pixel values, we want to apply an entropy coder. The most classical entropy coders are Huffman coding<sup>2</sup> [Huf52] and arithmetic coding<sup>2</sup> [Ris76]. Meanwhile, there are much

<sup>2</sup> sources from: <http://michael.dipperstein.com>

more sophisticated compression methods available that often rely on Huffman or arithmetic coding as part of their coding chain. Such compression methods are for example given by *gzip* (we use version 1.3.12) or *bzip2* (we use version 1.0.5).

One of the best compressors available so far are given by the *PAQ* data compressors [Mah05]. *PAQ* describes a whole family of lossless, GPL-licensed data compression archivers. They are based on a context mixing algorithm, which is related to prediction by partial matching (PPM) [Mof90]. More precisely, *PAQ* uses a predictor that is provided with a large number of models conditioned on different contexts, often even tuned to special file formats. This predictor is applied in combination with arithmetic coding. We consider the *PAQ806* release.

Unfortunately, the *PAQ806* compressor is rather slow. Therefore, we can also use the single file compressor *LPAQ* instead, more precisely the *LPAQ2* release. *LPAQ* is a modified version of *PAQ*, which is faster at the expense of compression.

Our codec allows to choose between all mentioned compressors. A comparison of their compression capabilities regarding our data is depicted in Table 3.2. For almost all examples, *PAQ* outperforms the other methods regarding the compression rate. However, it is also by far the slowest method. Huffman and arithmetic coding, as well as *gzip* and *bzip2* are up to small differences the fastest methods. Depending on the application, it is up to the user which entropy coder should be chosen. For highest compression, *PAQ* is recommended. For fastest compression, *bzip2* or *gzip* should be used. *LPAQ* is a reasonable trade-off between high quality compression and run time. Thus, we decided to set *LPAQ* as default pixel value encoder in our codec.

### 3.2.4 File Format

After this last step, we end up with two encoded data parts, namely the encoded edge image and the encoded grey or colour values. Including header data, our encoded image consists of the following parts:

- size of edge data, needed to split the encoded edge image from the encoded grey or colour values, (requires 4 bytes).
- edge image coder, indicating how the edge image has been encoded; default: *JBIG*, (3 bit).
- colour bit, indicating if a greyscale or colour image has been encoded, (1 bit).

**Table 3.2:** Comparison of different entropy coders regarding their compression ratio and time. Considered are six randomly chosen pixel value data files, created by our codec. Coders: Huffman coding (HC), Huffman coding using canonical codes (HCc), arithmetic coding with static model (ACs), arithmetic coding with adaptive model (ACa), *gzip* (Version 1.3.12), *bzip2* (Version 1.0.5), *LPAQ2*, and *PAQ8o6*. CPU: Intel Core 2 Duo T7500 @ 2.20Ghz. The best result for each file is in bold face letters.

coder	File					
	1	2	3	4	5	6
<b>file size in bytes:</b>						
original	12606	9006	29298	96063	13128	5514
HC	7058	4164	27398	48808	2178	6342
HCc	7160	4336	26380	48900	2420	5439
ACs	6944	4087	26714	48243	1491	5800
ACa	7066	4241	25972	42168	1710	5253
<i>gzip</i>	4300	2222	21655	8093	697	5223
<i>bzip2</i>	4016	2095	24163	7281	746	4981
<i>LPAQ2</i>	3112	1658	17974	5504	<b>467</b>	4300
<i>PAQ8o6</i>	<b>2842</b>	<b>1495</b>	<b>14766</b>	<b>4967</b>	468	<b>3949</b>
<b>encoding time in ms:</b>						
HC	3	<b>1</b>	4	<b>5</b>	1	1
HCc	3	2	4	6	< <b>1</b>	1
ACs	3	2	9	19	< <b>1</b>	1
ACa	11	3	16	57	6	4
<i>gzip</i>	< <b>1</b>	2	<b>3</b>	11	5	< <b>1</b>
<i>bzip2</i>	4	2	7	68	1	3
<i>LPAQ2</i>	42	31	86	168	29	39
<i>PAQ8o6</i>	1411	686	2470	9041	941	693
<b>decoding time in ms:</b>						
HC	< <b>1</b>	3	5	<b>5</b>	< <b>1</b>	2
HCc	4	4	19	26	< <b>1</b>	4
ACs	2	3	9	18	1	3
ACa	10	4	16	51	5	2
<i>gzip</i>	3	2	<b>3</b>	9	2	1
<i>bzip2</i>	< <b>1</b>	<b>1</b>	4	<b>5</b>	< <b>1</b>	< <b>1</b>
<i>LPAQ2</i>	37	34	89	190	29	34
<i>PAQ8o6</i>	1200	694	2438	8377	825	711

- entropy coder, indicating which entropy coder has been used; default: *LPAQ2*, (3 bit).
- parameter  $d_{tr}$  for recursive search of grey/colour values; default: 1, (1 byte).
- numbers of quantisation intervals  $q_*$ ; default: 25, (1 or 3 bytes).
- sampling distances  $d_*$ ; default: 10, (1 or 3 bytes).
- encoded edge image, (variable length, given in first 4 bytes of header).
- encoded pixel values, (variable length).

### 3.3 Decoding

The decoding process is now straight-forward. We start our decoding approach by reading the header information. We split the encoded edge image from the encoded grey or colour values and decode them with the corresponding decoders of the bi-level or entropy compression methods. On one hand, this gives us a binary edge image. Note that by this edge image, we also know the final image size. On the other hand, we get the quantised and subsampled grey or colour values. For those, we first revert the quantisation using Equation (3.3). Since we already know the edge image, we can now perform the pixel tracing algorithm again, as done during encoding (see Section 3.2.3). We redistribute the subsampled grey or colour values along the edge segments and fill in missing pixels by linear interpolation. The outcome is an image which contains the decoded colours on both sides of each edge (see Figure 3.1(b)). Those pixels now form the inpainting data such that the remaining grey or colour values can be reconstructed by homogeneous diffusion inpainting (Chapter 2).

**Remark** In the 1-D case, homogeneous diffusion inpainting is equivalent to a simple linear interpolation. In this sense, we first perform a 1-D homogeneous diffusion process along the edges when reconstructing the subsampled signal. Then we perform a 2-D homogeneous diffusion inpainting to recover pixels between edges. Using the 2-D homogeneous diffusion inpainting on the subsampled pixels directly would give results that are not nearly as good as the ones obtained by the successive application.



## 3.4 Experiments

Let us now investigate the capabilities of the suggested codec. To this end, we first give a comparison to well-established and state-of-the-art compression methods, namely *JPEG* and *JPEG2000*. We then want to identify the limitations of our method. For the quantitative comparison, we use the *peak-signal-to-noise ratio* (PSNR), a common error measure for the comparison between compressed images:

Let  $f_{\max}$  be the maximal possible pixel value, which is 255 in our case. Furthermore, let  $J$  be the set of all pixel indices, and  $(f_{*,i})_{i=1..N}$  and  $(u_{*,i})_{i=1..N}$  the pixel values of the original image in channel  $*$  and its reconstructed/decompressed version, respectively. The PSNR is defined via the *mean squared error* (MSE):

$$\text{PSNR}(\mathbf{u}, \mathbf{f}) := 10 \cdot \log_{10} \left( \frac{f_{\max}^2}{\text{MSE}(\mathbf{u}, \mathbf{f})} \right) [\text{dB}] , \quad (3.4)$$

with

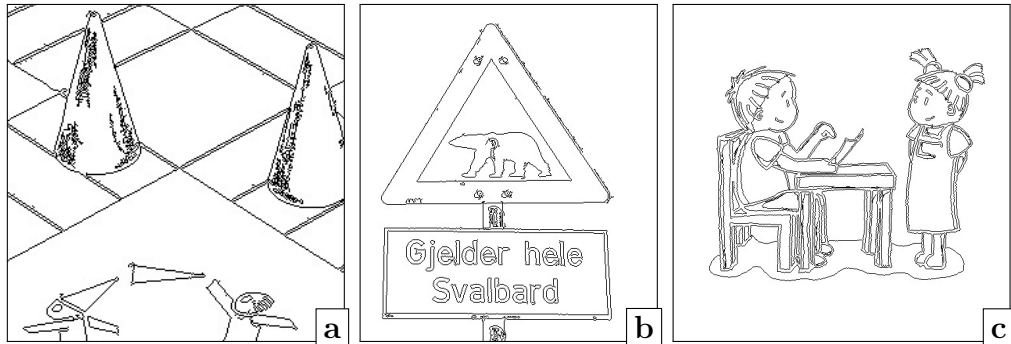
$$\text{MSE}(\mathbf{u}, \mathbf{f}) = \frac{1}{|J| \cdot |\{R,G,B\}|} \sum_{* \in \{R,G,B\}} \sum_{i \in J} (f_{*,i} - u_{*,i})^2 . \quad (3.5)$$

For grey valued images – having only one channel –  $|\{R, G, B\}|$  is replaced by 1 and the sum over the channels vanishes.

In the following experiments, all runtime measurements are based on one core of an Intel Core 2 Duo T7500 @ 2.20Ghz CPU and measure the elapsed time.

### 3.4.1 Comparison with JPEG and JPEG2000

For the comparison with the well-established *JPEG* standard and with the more advanced *JPEG2000* codec, we use the encoders provided by the image processing tool *imagemagick*. We fix all parameters of our codec to their default values (see Section 3.2.4). Figure 3.5 shows three different test images and their compressed versions using *JPEG*, *JPEG2000* and our codec. The underlying edge images used for our results are depicted in Figure 3.4. They were created by choosing parameters that give visually pleasant inpainting reconstructions. Moreover, we demand that the edge images look reasonable. That means we avoid false edges by choosing appropriate thresholds. A cropped detail for each result is shown in Figure 3.6. The compression rates for all images lie between 0.16 and 0.37 bits per pixel (bpp). This corresponds to compression ratios between roughly 145:1 and 65:1, provided the original colour images use 3 bytes per pixel. Besides the

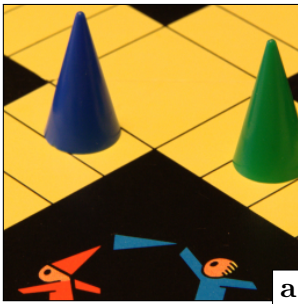

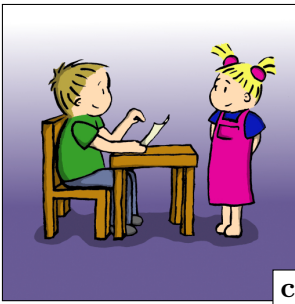
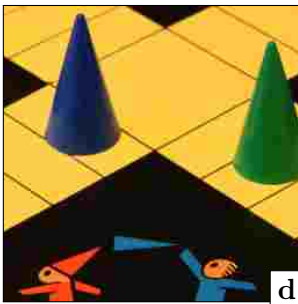


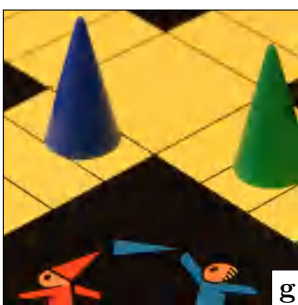


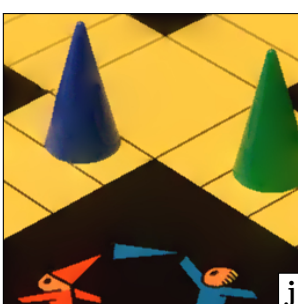




**Figure 3.4:** Edge images obtained by zero-crossings-based edge detection such that visually pleasant reconstructions are obtained. Edge detector parameters are chosen such that false edges are avoided. Edge image for (a) *coppit* ( $T_1 = 4.3$ ,  $T_2 = 23$ ,  $\sigma = 0.5$ ), (b) *svalbard* ( $T_1 = 3.5$ ,  $T_2 = 20$ ,  $\sigma = 0.6$ ) and (c) *comic* ( $T_1 = 20$ ,  $T_2 = 20$ ,  $\sigma = 0.47$ )

visual comparison, Table 3.3 summarises the corresponding quantitative comparison, and depicts the encoding and decoding times for all images.

The quantitative and the visual analysis illustrate that our approach can be better than *JPEG* and even *JPEG2000*. For the test image *comic* we observe a remarkable difference of more than 3 dB between *JPEG2000* and our approach. Also visually, crucial differences become apparent (see Figure 3.6). *JPEG* as well as *JPEG2000* suffer from severe ringing artefacts. These are a consequence of their quantisation step in the corresponding frequency/wavelet domains and the following inverse transforms. Moreover, *JPEG* applies the cosine transform in  $8 \times 8$  blocks and thus suffers from unpleasant block artefacts, as well as highly distorted edges. Our method in contrast stores edges explicitly and gives clean reconstructions. In addition, *JPEG* is not able to preserve the smooth gradient in the background of *svalbard* or *comic*. Our codec interpolates between the quantised colours so that the smooth gradient can be reconstructed almost perfectly. Thereby also for a greatly reduced grey or colour range, quantisation artefacts are hardly visible. Another drawback of the transform-based approaches can be discovered in the cropped detail image of *svalbard*. The screw cap within the bear completely vanishes for both *JPEG* as well as *JPEG2000*. Our approach stores the edges of such details, so that they are well preserved.

The previous results have been obtained with the default settings of our codec (see Section 3.2.4). By changing the parameters, we can influence the compression rate and quality of an image. Most obviously, we can get higher compression rates with larger sampling distances or fewer quantisa-

original			
JPEG			
PSNR	26.61	23.38	24.25
JPEG2000			
PSNR	28.13	27.68	26.77
edge-based			
PSNR	30.31	30.14	30.20

**Figure 3.5:** Comparison of compression methods for different test images at 0.37, 0.16 and 0.19 bits per pixel (bpp) (from left to right). **Columns from left to right:** *coppit* ( $256 \times 256$ , real world), *svalbard* ( $380 \times 431$ , real world), *comic* ( $512 \times 512$ , synthetic), **Rows from top to bottom:** original image, *JPEG*, *JPEG2000*, and our codec with default parameters and edge images as depicted in Figure 3.4.



**Figure 3.6:** Cropped detail ( $64 \times 64$ ) for each image depicted in Figure 3.5.

**Table 3.3:** Comparison of the PSNR and elapsed encoding/decoding time for the different test images (see Figure 3.5) and different compression methods. The best results are in bold face letters.

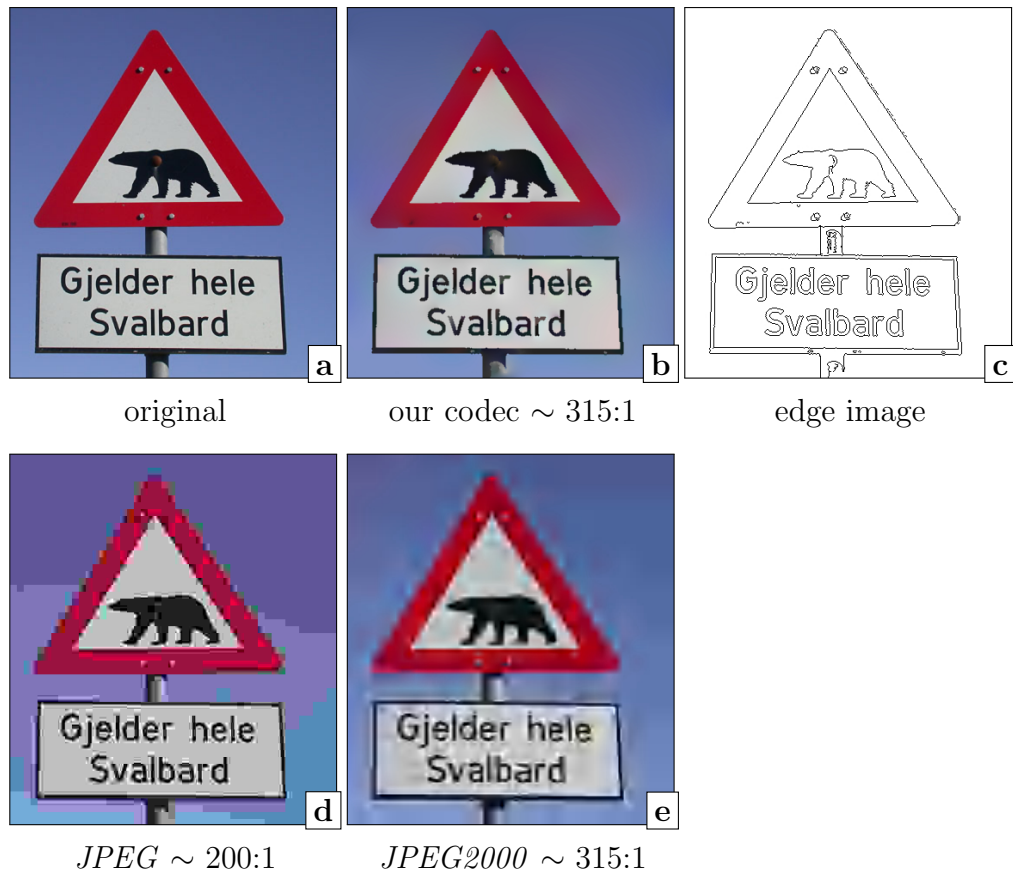
image	<i>coppit</i>	<i>svalbard</i>	<i>comic</i>
compression rate (bpp)	0.37	0.16	0.19
<b>error measure (PSNR in dB):</b>			
<i>JPEG</i>	26.61	23.38	24.25
<i>JPEG2000</i>	28.13	27.68	26.77
our method	<b>30.31</b>	<b>30.14</b>	<b>30.20</b>
<b>encoding time in ms:</b>			
<i>JPEG</i>	<b>8.66</b>	<b>13.80</b>	<b>19.34</b>
<i>JPEG2000</i>	57.51	93.47	164.42
our method	117.08	146.79	212.55
<b>decoding time in ms:</b>			
<i>JPEG</i>	<b>7.14</b>	<b>13.35</b>	<b>4.20</b>
<i>JPEG2000</i>	28.50	49.39	74.20
our method	172.78	316.94	458.40

tion intervals. Figure 3.7 demonstrates what is possible if this is carried to the extreme: We reduce the number of edge pixels by choosing  $T_1 = 15$  instead of 3.5. Thus, low contrast edges are not detected anymore and we obtain an edge image which contains only the visually most important edges (see Figure 3.7(c)). For each channel only 15 different values are used. The sampling distance along the edges is set to 45 and for the recursive search along edges  $d_{tr} = 10$ .

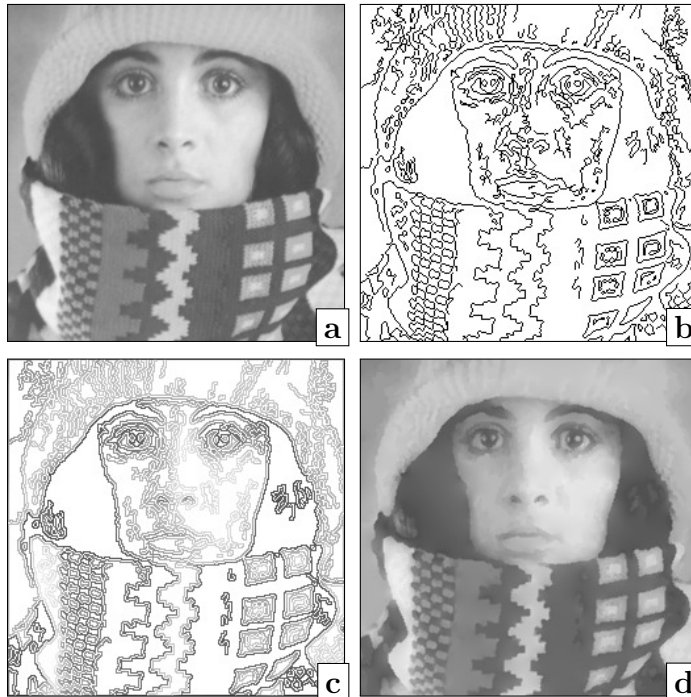
Using the lowest quality parameter provided by *imagemagick*, *JPEG* reaches only a compression ratio of 200:1. Its visual and quantitative quality (21.68 dB) is already far below the PDE-based result (28.18 dB). Moreover, our codec exhibits a compression ratio of 315:1.

In contrast to *JPEG*, for *JPEG2000* *imagemagick* is able to compress the image with a ratio of 315:1. With a PSNR of 23.09 dB, the quality of this image is better than the *JPEG* result but still not comparable to the result of our method. Considering the extreme compression rate, a PSNR of 28.18 dB for the our result is more than satisfactory.

As we have seen, the compression rate can also be influenced by the



**Figure 3.7:** Compression methods driven to the extreme: (a) Original image; (b) Our codec using the edge image as depicted in (c) and default settings except for  $q_* = 15$ ,  $d_* = 45$  ( $* \in \{R, G, B\}$ ) and  $d_{tr} = 10$  gives 0.08 bpp (i.e. approx. 315:1) with PSNR 28.18 dB; (c) Edge image obtained with zero-crossings based edge detector ( $T_1 = 15$ ,  $T_2 = 20$ ,  $\sigma = 0.6$ ); (d) *JPEG* with minimal quality parameter at 0.12 bpp (i.e. approx. 200:1) with PSNR 21.68 dB; (e) *JPEG2000* at 0.08 bpp (i.e. approx. 315:1) with PSNR 23.09 dB.



**Figure 3.8:** (a) Test image *trui* ( $256 \times 256$ ) (b) Edge image, obtained by zero-crossings-based edge detector ( $T_1 = 1.1$ ,  $T_2 = 2.6$ ,  $\sigma = 1.2$ ). (c) Reconstructed colours adjacent to the edges. (d) Inpainting result (PSNR of 30.16 dB); Compression rate of encoded image: 0.54 bpp.

underlying edge image. By choosing  $T_1 = 15$  instead of  $T_1 = 3.5$  we removed some edge pixels, yielding a higher compression rate. Note that these edge pixels should usually not be removed in order to guarantee quality. For cartoon-like images the corresponding edge image is up to small variations more or less unique.

Finally the compression rate can be increased by using *PAQ806* instead of *LPAQ2* at the expense of run time. Vice versa, a faster compression with lower compression rate is obtained when *LPAQ2* is replaced for example by *bzip2*.

### 3.4.2 Limitations

At the end of this section we briefly mention the limitations of our codec.

Obviously, our method is not well suited for images that contain a lot of textured areas. Figure 3.8 gives an example where our method is not competitive to conventional compression methods anymore. We detect so

many texture edges that our method requires too much storage to get results of reasonable quality. For this example and default settings our codec gives a compression rate of 0.54 bpp and a PSNR of 30.16 dB. At the same compression rate *JPEG* leads to a PSNR of 36.02 dB and *JPEG2000* to a PSNR of 37.24 dB.

Another limitation is the low robustness of our decompression algorithm against corrupted files which typically result from transmission channels being exposed to noise. Since pixel locations and colours are encoded separately, errors in either part might result in a wrong assignment of colours to mask points, or to pixels being reconstructed at arbitrary locations in the image. However, such problems can already be handled in the transmission of the file: Modern digital communication channels are typically equipped with fast error correction algorithms and retransmission mechanisms that assure file integrity.

### 3.5 Conclusion

In this chapter we have presented a conceptually simple, but highly efficient way to compress cartoon-like images. By extracting edges and adjacent pixel values, encoding them efficiently and using homogeneous diffusion for reconstruction, we have created a new codec, which can even outperform *JPEG2000*. Our results clearly indicate that cartoon-like images need a specialised treatment as offered by our codec. By storing edges explicitly, small details and sharp discontinuities are well preserved. Thereby, our codec is not only able to encode, but is also able to decode images in real time.



# Chapter 4

---

## Optimising Spatial and Tonal Data

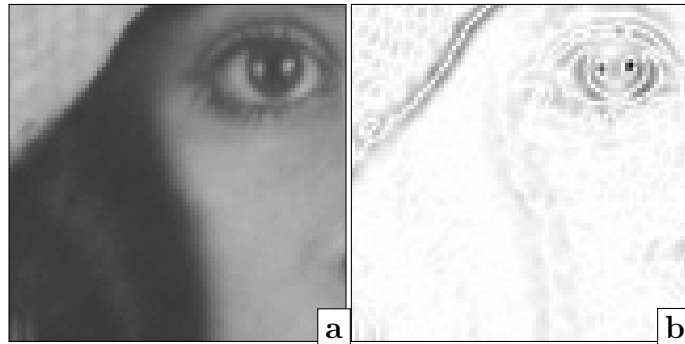
### 4.1 Motivation

In the previous chapter, we have presented a PDE-based image compression codec for cartoon-like images. In the spirit of second generation image coding, the selection of the location for the inpainting data was based on edges.

Apart from the fact that edges provide meaningful image features, this choice can also be motivated by the optimality of the chosen data corresponding to the theory of Belhachmi et al. [BBBW09]. As already mentioned in Section 3.1, they present a continuous analysis on spatially optimal data selection for homogeneous diffusion interpolation. Their framework is based on the theory of shape optimisation and suggests to choose a pixel density that is an increasing function of the modulus of the image Laplacian. Thus, the selection of inpainting data for cartoon-like images adjacent to edges (cf. Figure 3.2) is justified. However, as it is illustrated in in Figure 4.1, for blurry edges the interpolation data would apparently need to be spread more smoothly and with a larger distance to each side of the edges.

Thus, although features like edges can be perceptually relevant, one cannot expect that they are in general optimal w.r.t. some error norm. This might be another reason for the bad performance of the edge-based codec regarding natural images (cf. Section 3.4.2).

As a result, this raises the fundamental question of which data is actually optimal, and how to determine it. In order to make the optimality



**Figure 4.1:** (a) Zoom into test image *trui* ( $256 \times 256$ , real world). (b) Modulus of the Laplacian of (a) normalised to  $[0,255]$ . The darker the colour, the larger the modulus of the Laplacian.

result of Belhachmi et al. applicable to the practically relevant discrete setting, they suggest to apply dithering techniques. However, those can introduce additional errors. Moreover, the continuous optimality result only holds for homogeneous diffusion inpainting, but not for more sophisticated differential operators.

Another aspect that is often forgotten in PDE-based image compression, is the fact that not only the spatial data, but also the tonal data can be selected freely. Thus, unlike in pure inpainting, we are not bound to the grey or colour values as given in the original image. The question for optimal tonal data is as valid as the one for optimal spatial data.

The goal of this chapter is to address the aforementioned aspects. We show how optimised spatial and tonal data can be obtained for the discrete inpainting problem. In this process, we specify a fixed amount of freely distributed inpainting data. As the amount of pixels approximately correlates with a specific compression ratio, this is also expedient with respect to image compression.

We first develop our methods using the simplest PDE, homogeneous diffusion inpainting, as it was done in [MHW<sup>+</sup>12]. This allows us to show the potential behind our approach, as homogeneous diffusion usually has a bad reputation for inpainting tasks. Then, we extend the concepts towards the biharmonic operator and edge-enhancing diffusion.

Note that the optimality in our results in this framework solely refers to the reconstruction quality, but only indirectly includes coding costs via the amount of pixels. However, it is well known that specific pixel distributions can be encoded more efficiently than others. Examples are tree-based codecs such as the ones that have been mentioned in Chapter 1. Thus, the results

in this chapter constitute only a first step towards an optimal data selection in PDE-based image compression. Nevertheless, we will see that even with those methods, it is already possible to beat the quality of *JPEG2000*.

One should also note that we did not optimise our algorithms with respect to runtime, as we regard it as a proof-of-concept only. Thus, our methods can require several hours to days to process typical images.

As a response to our work, Hoeltgen et al. have proposed an energy-based approach to optimise the interpolation data for homogeneous diffusion inpainting [HSW13]. The solution of the resulting optimal control problem is approximated by solving a series of simpler convex optimisation problems. As a result, their approach gives slightly better results than our method, and is considerably faster. In [CRP14], Chen et al. have extended this framework to make it applicable for inpainting with the biharmonic operator as well as TV regularised inpainting. In contrast to biharmonic inpainting, TV regularised inpainting has not given considerably better results than inpainting with homogeneous diffusion. In contrast to those methods that need the differential operators to meet specific requirements, our spatial optimisation approaches can be applied to any inpainting method out of the box. Moreover, we can easily impose further requirements to a mask, for instance demanding that specific mask pixels are not included into the optimisation framework. This is particularly useful if optimised data is included into existing codecs as we will do it in Chapter 5.

Let us briefly mention more related work before we dive deep into our algorithms. From the Green function of the Laplace operator, it follows that homogeneous diffusion inpainting involves radial basis functions. These functions are popular for scattered data interpolation, and some of them have also been used for the inpainting of corrupted images [DFTT11, US05]. However, such problems usually do not allow to optimise the location and the grey values of the inpainting data set.

The holographic image representation presented in [BHN98] maps the image into a sequence of sample pixels, such that any partition of this sequence allows for a reconstruction of the whole image with similar quality. This requires the samples in each portion to be equally optimal. On the contrary, our goal is to reduce the image to only one set of optimal samples.

The idea to approximate an image by identifying significant pixels has also been investigated in several papers. In [ELPZ97] this concept is used for progressive sampling. Similarly [DFI02], introduce an adaptive thinning strategy based on Delaunay triangulations for the sampling of scattered data. In [DI04] and [DDI06], they improve this strategy for image compression.

Finally, the problem of optimising the sampling of data with fixed re-

construction method is also known in the spline literature as so-called *free knot problem* [dR68].

This chapter is organised as follows: We start Section 4.2 with the spatial optimisation problem for the case of homogeneous diffusion inpainting. To this end we first review the analytic approach of Belhachmi et al. [BBBW09]. Then we present two approaches that are applied sequentially to optimise the pixel locations: a probabilistic sparsification method, followed by a non-local pixel exchange. At each step we evaluate the corresponding method and compare it to the previous ones. In the second part of this chapter, in Section 4.3, we show how the results can be improved further by an exact optimisation of the tonal data. In Section 4.4 we explain how the optimisation algorithms can be generalised for inpainting with the biharmonic operator and EED, respectively. Afterwards, in Section 4.5 we evaluate the potential of the optimisation concepts for PDE-based image compression. We conclude the chapter in Section 4.6.

## 4.2 Optimising Spatial Data

### 4.2.1 The Analytic Approach

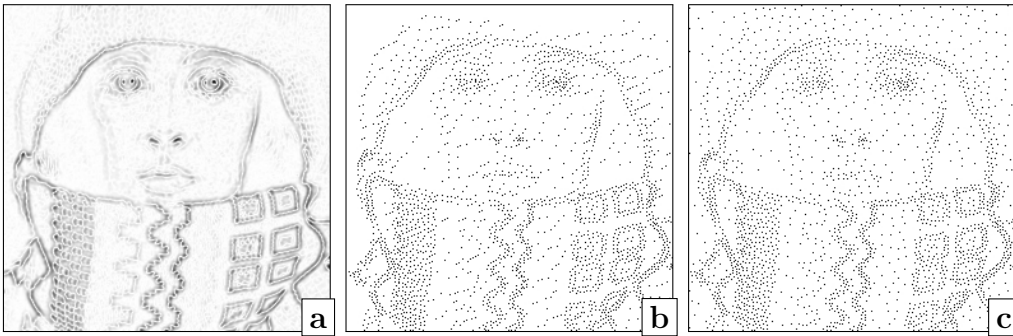
In order to answer the question about the optimal data selection, Belhachmi et al. use the sophisticated mathematical theory of shape optimisation. By investigating different models for homogeneous diffusion interpolation, it could be proven that the density distribution of the data points should be chosen as an increasing function of the Laplacian magnitude of the original image  $|\Delta f|$ . However, this true optimality result returns a continuous density function rather than a discrete pixel mask.

In order to obtain a method that is applicable to the discrete setting, we follow the steps suggested in the original paper: We first apply a small amount of Gaussian presmoothing with standard deviation  $\sigma$  and obtain  $f_\sigma$ . As stated in the paper, this is a common procedure in image analysis to address the ill-posedness of differentiation. Then we compute the Laplacian magnitude  $|\Delta f_\sigma|$  using finite differences and rescale it such that its mean represents the desired point density, given as fraction  $\theta$  of all pixels. Finally, any dithering algorithm that preserves the average grey value can be applied to obtain the binary point mask.

In the original paper, the widely used error diffusion method of Floyd and Steinberg [FS76] is used. However, as depicted in Figure 4.3 different dithering methods result in different discrete point masks, and eventually in reconstructions of different quality. In this thesis we favour the sophisticated



**Figure 4.2:** Test images: (a) *trui* ( $256 \times 256$ ), (b) *walter* ( $256 \times 256$ ), (c) *peppers256* ( $256 \times 256$ ).



**Figure 4.3:** (a) Smoothed Laplacian magnitude of the test image *trui* (see Figure 4.2) using  $\sigma = 1$  (rescaled and inverted). (b, c) Dithered versions of (a) using Floyd-Steinberg error diffusion and electrostatic halftoning, respectively. Using (b) and (c) as mask for homogeneous diffusion inpainting we obtain an MSE of 138.98 and 101.14, respectively. The masks have a density of 4%.

electrostatic halftoning [SGBW10] over simpler dithering approaches, since it has proven to be state-of-the-art for discretising a continuous distribution function, particularly in this specific scenario [Sch12]. Nevertheless, even with this method, dithering introduces errors. It remains an open question if it is the most suitable approach to discretise the continuous optimality result.

Another unsolved problem is that the theory demands the data points to be chosen as an increasing function of  $|\Delta f|$ . Yet, the optimal increasing function is still unknown. So far, we simply used the identity function of the Laplacian magnitude. As a remedy, we add now a parameter  $s > 0$  and dither  $|\Delta f_\sigma|^s$  instead. This choice is also motivated at the end of Section 6 in the original paper and allows to tune the density of the selected points

in homogeneous regions. The complete method, which we call the *analytic approach*, is summarised as follows:

---

**Algorithm: Analytic approach**

---

**Input:**

Original image  $f$ , Gaussian standard deviation  $\sigma$ , exponent  $s$ , desired pixel density  $\theta$ .

**Compute:**

1. Perform Gaussian presmoothing with standard deviation  $\sigma$ :  $f_\sigma = K_\sigma * f$ .
2. Compute  $|\Delta f_\sigma|^s$ .
3. Rescale  $|\Delta f_\sigma|^s$  to  $\frac{\theta \cdot f_{\max}}{\text{mean}(|\Delta f_\sigma|^s)} \cdot |\Delta f_\sigma|^s$  where  $f_{\max}$  is the maximal possible grey value.
4. Apply electrostatic halftoning to get  $\mathbf{C}$ .

**Output:**

Discrete pixel mask  $\mathbf{C}$ .

---

In order to evaluate the analytic approach let us apply it on the three test images *trui*, *walter* and *peppers256* (see Figure 4.3). Out of  $256 \cdot 256 = 65,536$  pixels we want 2601 mask pixels. That corresponds roughly a mask pixel density of 4% of all pixels and hence  $\theta = 0.04$ . As underlying error measure we use the *mean squared error (MSE)* (see Equation (3.5)). The parameters  $\sigma$  and  $s$  are chosen such that the MSE of the reconstruction becomes minimal ( $\sigma = 1.6$  and  $s = 0.8$ ). The Figures 4.7, 4.9 and 4.11 show the resulting masks (a) as well as the corresponding reconstructions (b). For comparison, we choose the same amount of pixels once randomly and once on a rectangular grid. In addition we consider a hexagonal grid that has due to its structure slightly more mask pixels (2612 instead of 2601). The resulting masks (a,d,g) as well as the reconstructions (b,e,h) are depicted in Figures 4.6, 4.8 and 4.10. Clearly, we can see that the visual as well as the quantitative reconstruction quality highly benefits from a dedicated point selection as done in the case of the analytic approach.

Note that in contrast to the following approaches, the analytic approach does not need to reconstruct the image over and over again. Therefore, it is realtime-capable, provided a fast dithering method is used. However, rather than on speed, the focus is here on the maximum possible quality

when spatial and tonal data can be chosen freely. The following methods can require several hours to days to process typical images. We are confident that this runtime can be significantly reduced, e.g. by exploiting parallel architectures. This is part of our ongoing research and is already tackled in [HSW13].

### 4.2.2 Probabilistic Sparsification

The analytic approach is a first suggestion how to choose the spatial data in a digital image. However, due to the mentioned sources of error this data might actually not be optimal yet. Since one of those sources of error concerns the discretisation of the continuous theory, it could justifiably be asked whether optimisation strategies that ground directly on the discrete inpainting formulation (2.8) could outperform the previous result.

Therefore, we refrain from the continuous formulation and investigate discrete optimisation strategies. The good news in the discrete setting is that the number of possible solutions is finite. Thus, there must be a pixel mask for a desired density that minimises the reconstruction error and we could find this mask by simply trying all possible solutions. However, already for an image of size  $256 \times 256$  pixels and a desired pixel density of 4% there are  $\binom{65536}{2601} \approx 4.65 \cdot 10^{4749}$  possible solutions. To overcome this combinatorial problem we introduce a method which we call *probabilistic sparsification*.

Let  $\mathbf{f}$  be a discrete given image and let  $\mathbf{r}(\mathbf{C}, \mathbf{f})$  be the function, which computes the solution  $\mathbf{u}$  of the discrete homogeneous inpainting process (2.9), depending on a mask  $\mathbf{C}$ :

$$\mathbf{r}(\mathbf{C}, \mathbf{f}) := \mathbf{u} = \mathbf{M}^{-1} \mathbf{C} \mathbf{f}. \quad (4.1)$$

The objective is to find the pixel mask  $\mathbf{C}$ , which marks a given fraction  $\theta$  of all pixels and minimises the  $\text{MSE}(\mathbf{u}, \mathbf{f})$  (see Equation (3.5)).

Starting with a dense mask, i.e. any pixel is a mask pixel, probabilistic sparsification removes iteratively the least significant mask pixels until a desired density is reached. In each step candidate pixels are selected and removed from the mask. The resulting sparser mask is used for inpainting. The significance of a candidate pixel can then be estimated by computing the local error, i.e., taking the squared difference of inpainted and original image in this pixel. Thus, we remove the candidates which exhibit the smallest error permanently from the mask and put back the remaining ones. A detailed description of our algorithm is given as follows:

**Algorithm: Probabilistic sparsification**

---

**Input:**

Original image  $\mathbf{f}$ , fraction  $p_{\text{cand}}$  of mask pixels used as candidates, fraction  $p_{\text{rm}}$  of candidate pixels that are removed in each iteration, desired pixel density  $\theta$ .

**Initialisation:**

$\mathbf{C} := \text{diag}(1, \dots, 1)^\top$ , thus  $K = J$ .

**Compute:**

While  $|K| > \theta \cdot |J|$  do

1. Choose randomly a candidate set  $T$  of  $\max(1, \lceil p_{\text{cand}} \cdot |K| \rceil)$  pixel indices from  $K$ , where  $\lceil \cdot \rceil$  denotes rounding to the closest integer.
2. For all  $i \in T$  reassign  $c_i := 0$ .
3. Compute  $\mathbf{u} := \mathbf{r}(\mathbf{C}, \mathbf{f})$ .
4. For all  $i \in T$  compute the local error  $e_i = (u_i - f_i)^2$ .
5. For all  $i$  of the  $\max(1, \lceil (1 - p_{\text{rm}}) \cdot |T| \rceil)$  largest values of  $\{e_i \mid i \in T\}$ , reassign  $c_i := 1$ .
6. Remove from  $K$  the indices  $i \notin T$  and clear  $T$ .

**Output:**

Pixel mask  $\mathbf{C}$ , s.t.  $\sum_{i \in J} c_i = \theta \cdot |J|$ .

---

The parameters  $p_{\text{cand}}$  and  $p_{\text{rm}}$  determine the fraction of mask pixels used as candidates and the fraction of candidates that are removed in each step, respectively. The larger  $p_{\text{cand}}$  and  $p_{\text{rm}}$  are chosen, the faster the algorithm converges: In each step around  $p_{\text{cand}} \cdot p_{\text{rm}} \cdot |K|$  pixels are removed. After  $k$  steps there are about  $(1 - p_{\text{cand}}p_{\text{rm}})^k \cdot |J|$  mask pixels left. Hence, for a density  $\theta$ , the algorithm terminates after roughly  $\log_{(1-p_{\text{cand}}p_{\text{rm}})} \theta$  iterations. In the worst case in each iteration only one pixel is removed. Thus, a lower bound is given by  $1 - \theta$  iterations.

Since there is a global interdependence between selected mask pixels, probabilistic sparsification does not guarantee to deliver optimal solutions but only approximate ones. It is a valid question, how the parameters  $p_{\text{cand}}$  and  $p_{\text{rm}}$  influence the quality of the resulting mask. Moreover, it is of interest, how much the quality of the results depends on the selected seed of the random number generator.



**Table 4.1:** Influence of the parameters  $p_{\text{cand}}$  and  $p_{\text{rm}}$  of probabilistic sparsification. In total, there have been 100 runs for each pair  $(p_{\text{cand}}, p_{\text{rm}})$  on the test image *trui* with desired pixel density  $\theta = 0.04$ . Numbers in the table are the mean and standard deviation of the MSE.

$p_{\text{rm}}$	$p_{\text{cand}}$			
	0.01	0.02	0.05	0.1
0.000001	77.6 $\pm$ 1.40	67.7 $\pm$ 1.40	<b>66.1 <math>\pm</math> 1.36</b>	70.7 $\pm$ 1.76
0.001	77.4 $\pm$ 1.33	67.8 $\pm$ 1.26	66.3 $\pm$ 1.41	70.5 $\pm$ 1.67
0.01	81.8 $\pm$ 1.68	73.0 $\pm$ 1.44	68.9 $\pm$ 1.81	69.4 $\pm$ 1.84
0.02	85.3 $\pm$ 1.71	76.4 $\pm$ 1.78	71.4 $\pm$ 1.56	70.6 $\pm$ 1.83
0.05	91.6 $\pm$ 1.82	82.7 $\pm$ 2.02	77.1 $\pm$ 2.03	74.6 $\pm$ 1.79
0.1	97.7 $\pm$ 2.14	89.5 $\pm$ 2.00	83.9 $\pm$ 2.22	80.8 $\pm$ 2.24

In order to answer those two questions, we run several experiments with different  $p_{\text{cand}}$  and  $p_{\text{rm}}$  values. The results for the test image *trui* are depicted in Table 4.1. Note that the candidate set as well as the set of pixels that are going to be removed is 1 if  $p_{\text{cand}}$  or  $p_{\text{rm}}$  would lead to sets smaller than 1 (cf. Step 1 and 5 of the algorithm).

Intuitively one would expect that the best results can be achieved by choosing  $p_{\text{cand}}$  as large as possible whereas  $p_{\text{rm}}$  is kept very small. In this case, we evaluate the quality of many mask pixels but remove only the very worst of them. However, there is an error in this reasoning. The measured error in each candidate pixel is referring to the combined removal of all candidates, not only the currently considered one. Thus, the error gives only an approximative measure of the significance of this pixel. The larger the candidate set is, the less accurate is this error estimate.

On the other hand,  $p_{\text{cand}}$  should also not be chosen too small, since it might then happen that the candidate set contains a lot of important mask pixels. As a consequence, it is more likely that important pixels are removed.

In contrast, the parameter  $p_{\text{rm}}$  should in general be chosen as small as possible, i.e., such that only one candidate is removed in each iteration, here  $p_{\text{rm}} = 0.000001$ . With larger values for  $p_{\text{rm}}$ , the probability that we remove important pixels increases. Only in the case where the candidate set is too large and thus results become unreliable, it might pay off to optimise  $p_{\text{rm}}$  as well.

Another observation of the results given in Table 4.1 is the robustness of the algorithm: The standard deviation does not exceed a value of 2.3 and

is even smaller for optimal  $p_{\text{cand}}/p_{\text{rm}}$ -values. This means that although the obtained masks for different seeds differ usually in a big part of the selected pixels, we still get qualitatively comparable results.

In Figure 4.7, Images (d) and (e) show the results of the probabilistic sparsification for the test image *trui* with a mask pixel density of 4% of all pixels, i.e.,  $\theta = 0.04$ . Since we are interested in the maximal possible quality we set  $p_{\text{cand}} = 0.2$  and  $p_{\text{rm}} = 0.000001$  (cf. Table 4.1). Both the visual as well as the quantitative results clearly outperform the ones of the analytic approach. This can also be observed for the other two test images *walter* and *peppers256* (see Figure 4.9 and Figure 4.11, Images (d) and (e),  $p_{\text{cand}}$  and  $p_{\text{rm}}$  have been optimised for each image individually). Thus, our assessment is confirmed that even though Belhachmi et al. offer a theory for the optimal selection of data in a continuous sense, yet the analytic approach is a suboptimal discretisation of this theory.

### 4.2.3 Non-local Pixel Exchange

As we have seen in the previous section probabilistic sparsification already outperforms the analytic approach. Still, it is not guaranteed to deliver the real optimum. Indeed, an obvious drawback of probabilistic sparsification is the fact that once a point is removed, it will never be put back again into the mask. Thus, especially at later stages, where only few mask pixels are left, important points might be removed. Moreover, due to the interdependence of the mask pixels, a point that was removed earlier might actually be useful to be set again at a later stage. The approach presented in this section, the so-called non-local pixel exchange is a remedy to this problem: It starts with a sparse, possibly suboptimal mask which contains already the desired density  $\theta$  of mask pixels. In each step it iteratively chooses a set of  $m$  randomly selected non mask pixels as candidates. The candidate which exhibits the largest local error is then exchanged with a randomly chosen mask pixel. If the inpainting result with the new mask is worse than before, we revert the exchange. Otherwise we proceed with the new mask. Therefore, the non-local pixel exchange can only improve the result. In detail the algorithm reads:

**Algorithm: Non-local pixel exchange**

---

**Input:**

Original image  $\mathbf{f}$ , (pre-optimised) pixel mask  $\mathbf{c}$  (and thus  $K$ ), size  $m \leq |K|$  of candidate set

**Initialisation:**

$\mathbf{u} := \mathbf{r}(\mathbf{C}, \mathbf{f})$  and  $\mathbf{C}^{\text{new}} := \mathbf{C}$ .

**Compute:**

Repeat

1. Choose randomly a candidate set  $T$  of  $m$  pixel indices from  $J \setminus K$ .
2. For all  $i \in T$  compute the error  $e_i = (u_i - f_i)^2$ .
3. Exchange step:  
Choose randomly a index  $j$  from  $K$  and set  $c_j^{\text{new}} := 0$ .  
Set  $c_i^{\text{new}} = 1$  for the  $i$  of the largest value of  $\{e_i \mid i \in T\}$ .
4. Compute  $\mathbf{u}^{\text{new}} := \mathbf{r}(\mathbf{C}^{\text{new}}, \mathbf{f})$ .
5. If  $\text{MSE}(\mathbf{u}, \mathbf{f}) > \text{MSE}(\mathbf{u}^{\text{new}}, \mathbf{f})$   
     $\mathbf{u} := \mathbf{u}^{\text{new}}$  and  $\mathbf{C} := \mathbf{C}^{\text{new}}$ .  
    Update  $K$ .
- Else  
        Reset  $\mathbf{C}^{\text{new}} := \mathbf{C}$ .
6. Clear  $T$ .

until no exchange-pairs can be found that improve the result  
(or alternatively until a desired number of repetitions is reached).

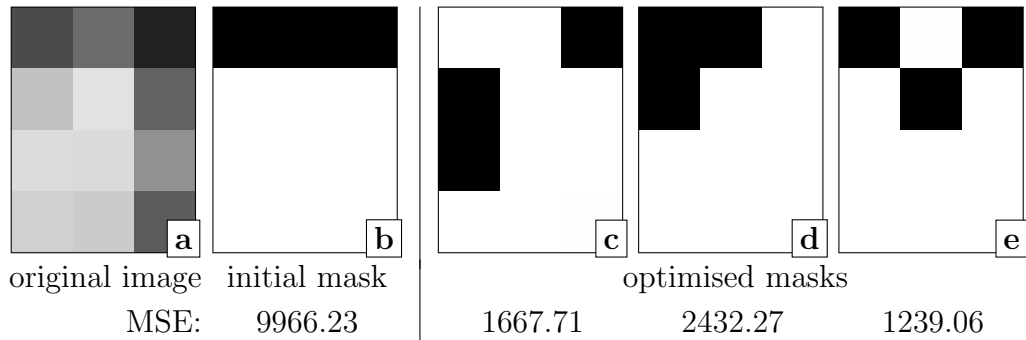
**Output:**

Post-optimised pixel mask  $\mathbf{C}$ .

---

Because at each iteration we always exchange only one candidate pixel, it is clear that the approach is not equivalent to an exhaustive search through all possible combinations. Thus, we cannot guarantee convergence towards the global minimum.

Consequently, we are interested in an optimal parameter selection. It is a fallacy to think that larger candidate sets (i.e. larger values for  $m$ ) would lead to better solutions. This becomes evident if we set  $m$  to the maximal possible value, i.e.  $m = |J \setminus K|$ . Then, when performing the exchange, the algorithm always picks out of *all* mask pixels the one that



**Figure 4.4:** (a) Test image of size  $2 \times 3$ . (b) Initial mask with three mask pixels. (c-e) Different solutions after the application of the non-local pixel exchange with  $m = 1$  depending on which pixels are randomly exchanged.

depicts the largest error. Eventually, the solution cannot be improved, no matter which mask pixel is exchanged. However, at this stage it cannot be ruled out that the exchange with a different non-mask pixel depicting a smaller error improves the solution further. This is due to local nature of our error measurement which is not necessarily giving a reliable estimate for the actual significance of a pixel position.

Choosing the other extreme, i.e.  $m = 1$ , it seems we are on the safe side. We randomly exchange in each iteration only one mask pixel by a non-mask one. Since this is repeated until the solution does not improve any further, we expect that in this case the algorithm converges towards an optimal solution in terms of a two-pixels-exchange. However, this does not hold as well: Since each exchange has a global impact, it might happen that an exchange leads to a mask that cannot further be improved by additional exchanges. However, another previous exchange, would have maybe allowed further improvements. In other words, we can get stuck in local minima. Figure 4.4 illustrates this problem. Depending which pixels are randomly selected and exchanged, we obtain different solutions (c-e) for the given test image (a) and the initial mask (b). Moreover, the results are different in quality. The right most solution (e) corresponds to the real optimum. As a consequence, we can also conclude that the solutions by the non-local pixel exchange depend on the initial mask.

Table 4.2 shows the results for different choices of  $m$  when the non-local pixel exchange is applied for 500,000 iterations to the masks of the test image *trui* from the previous sections (randomly selected, rectangular grid, hexagonal grid (see Figure 4.6(a,d,g)) and analytic approach, probabilistic sparsification (see Figure 4.7(a,d))).

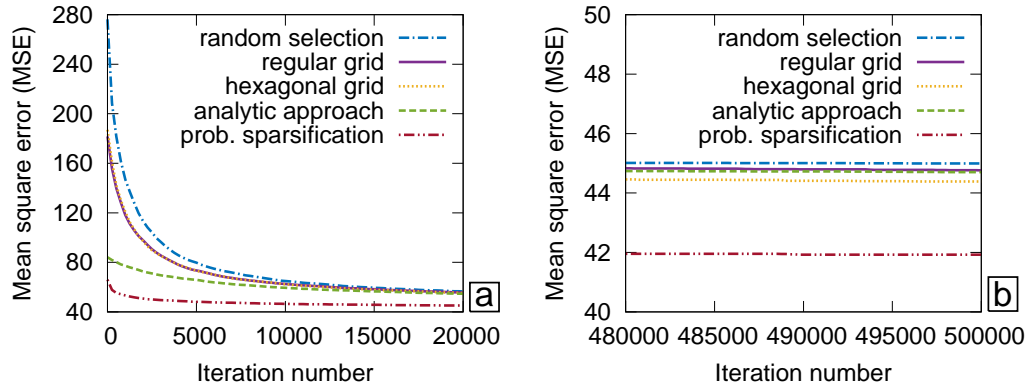
**Table 4.2:** Mean squared error after 500,000 iterations for different values for  $m$ , when the non-local pixel exchange is applied to different masks of the *trui* test image (see Figure 4.6(a,d,g) and Figure 4.7(a,d)). The best result for each mask is marked in boldface.

mask	$m$							
	1	5	10	20	30	40	50	100
randomly selected	49.88	46.43	44.99	<b>44.78</b>	45.00	45.07	45.25	48.22
rectangular grid	49.67	45.79	45.19	<b>44.76</b>	45.16	45.56	45.76	48.33
hexagonal grid	49.13	45.91	45.01	44.99	<b>44.39</b>	45.41	45.27	47.54
analytic approach	49.19	45.88	45.14	<b>44.70</b>	45.33	45.56	46.13	49.31
probabilistic sparsification	43.72	42.45	42.34	<b>41.92</b>	41.97	42.49	42.19	43.65

As expected, our results confirm that neither very small values for  $m$  nor large values give the optimal results. In this case, the best choice for  $m$  is given by a value around 10 to 30. It seems that restricting the size of  $m$  to these values, adds a moderate degree of randomness which allows us to escape from local minima. Moreover, we observe that for most of the masks we end up after 500,000 iterations with masks of comparable quality (MSE lies in the range of 41 to 45). Thus, the algorithm is not too sensitive to the initialisation.

To investigate the influence of the initialisation more closely, let us consider the convergence behaviour of the non-local pixel exchange, illustrated in Figure 4.5. For each mask we choose the optimal values for  $m$ .

Our first observation is that the method is able to improve any mask significantly, especially within the first few iterations (see Graph (a)). After 500,000 iterations there are only minor improvements (see Graph (b)). The best result with an MSE of 41.92 is obtained with the mask from probabilistic sparsification. Moreover, when using this mask, after only 3,000 iterations the MSE is already below 50. Using the other masks we need around 50,000 iterations to reach a comparable quality. This demonstrates the benefits we gain from the probabilistic sparsification. Another observation is that, even though after 500,000 iterations the improvements are minor, still the quality slightly enhances. Thus, the method actually did



**Figure 4.5:** Convergence behaviour when the non-local pixel exchange is applied to different masks (cf. Figure 4.6(a,d,f), Figure 4.7(a,d)) of the *trui* test image with optimal parameter  $m$  (cf. Table 4.2). (a) first 20,000 iterations; (b) last 20,000 iterations up to 500,000.

not converge yet. Although we found a good solution, it is not the optimal one. This assessment is supported by the fact that there is still a quality gap between the results obtained by the different masks. Another initialisation could potentially even yield better results.

Figure 4.7(g) depicts the mask that allows the so far best reconstruction (see Figure 4.7(h)) for *trui*, namely the one which was obtained by applying the non-local pixel exchange to the result obtained by probabilistic sparsification. The MSE improved from 66.11 to 41.92. This improvement is also reflected by comparing the images visually. As for the previous methods, Figure 4.9(g,h) and Figure 4.11(g,h) provide the results for the test images *walter* and *peppers256*, supporting the made observations ( $m$  has been optimised for each image individually).

### 4.3 Optimising Tonal Data

So far, all the optimisation approaches only proposed a solution for the spatial optimisation problem. For the reconstruction, we use the original grey values of the input image at the selected locations. However, there is no reason that would prevent us from using different grey values at those positions. In contrast, we may accept to introduce some error at the positions of mask pixels in favour of a lower overall reconstruction error. From a data compression perspective, changing the grey value will not increase the amount of data that needs to be stored but possibly allows a significant gain in quality. In this section we present an approach that allows us to determine the optimal grey values exactly for any given mask.

#### 4.3.1 Least Squares Approximation

In order to find the optimal grey values  $\mathbf{g}$  for a given mask  $\mathbf{C}$ , we consider the following minimisation approach:

$$\arg \min_{\mathbf{g}} \|\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g})\|^2, \quad (4.2)$$

where  $\|\cdot\|$  denotes the euclidean norm,  $\mathbf{f}$  denotes as before the original image and  $\mathbf{r}(\mathbf{C}, \mathbf{g})$  (see Equation (4.1)) the function, which computes the solution of the discrete homogeneous inpainting process. Since  $\mathbf{r}$  is a linear function in  $\mathbf{g}$  we can decompose the reconstruction  $\mathbf{r}(\mathbf{C}, \mathbf{g})$  as follows:

Let  $\mathbf{e}_i$  denote the  $i$ -th canonical basis vector of  $\mathbb{R}^{|J|}$ , i.e. the vector with a 1 in the  $i$ -th coordinate and zeros elsewhere. Then we call  $\mathbf{b}_i = \mathbf{r}(\mathbf{C}, \mathbf{e}_i)$  the *inpainting echo* of the  $i$ -th pixel and we obtain:

$$\mathbf{r}(\mathbf{C}, \mathbf{g}) = \mathbf{r}\left(\mathbf{C}, \sum_{i \in J} g_i \mathbf{e}_i\right) = \sum_{i \in J} g_i \mathbf{r}(\mathbf{C}, \mathbf{e}_i) = \sum_{i \in J} g_i \mathbf{b}_i. \quad (4.3)$$

Note that the inpainting echo of non-mask pixels is  $\mathbf{0}$ , i.e.  $\mathbf{b}_i = \mathbf{0}$  if  $i \in J \setminus K$ :

$$\mathbf{b}_i = \mathbf{r}(\mathbf{C}, \mathbf{e}_i) = \mathbf{M}^{-1} \mathbf{C} \mathbf{e}_i \stackrel{i \in J \setminus K}{=} \mathbf{M}^{-1} \mathbf{0} = \mathbf{0}. \quad (4.4)$$

Thus, we can simplify Equation (4.3) to

$$\mathbf{r}(\mathbf{C}, \mathbf{g}) = \sum_{i \in K} g_i \mathbf{b}_i. \quad (4.5)$$

For our minimisation problem (4.2), this means that  $g_i$  can be chosen arbitrarily if  $i \in J \setminus K$ . The remaining  $g_i$  with  $i \in K$  can be obtained by

considering the least squares problem:

$$\arg \min_{\mathbf{g}_K} |\mathbf{B}_K \mathbf{g}_K - \mathbf{f}|^2, \quad (4.6)$$

where  $\mathbf{g}_K = (g_i)_{i \in K}$  is a vector of size  $|K|$ , and  $\mathbf{B}_K$  is a  $|J| \times |K|$  matrix which contains the vectors  $\mathbf{b}_i$ ,  $i \in K$  as columns.

The associated normal equations are given by

$$\mathbf{B}_K^\top \mathbf{B}_K \mathbf{g}_K = \mathbf{B}_K^\top \mathbf{f}. \quad (4.7)$$

The matrix  $\mathbf{B}_K^\top \mathbf{B}_K$  needs to be invertible, so that the solution of the normal equations is also a solution of the least squares problem. This, however, can easily be proven: It is sufficient to show that the column vectors of  $\mathbf{B}_K$ , i.e. all  $\mathbf{b}_i$  with  $i \in K$ , are linearly independent. It holds that (cf. Equation (4.4))

$$\mathbf{b}_i = \mathbf{r}(\mathbf{C}, \mathbf{e}_i) = \mathbf{M}^{-1} \mathbf{C} \mathbf{e}_i \stackrel{i \in K}{=} \mathbf{M}^{-1} \mathbf{e}_i. \quad (4.8)$$

Hence, for  $i \in K$ ,  $\mathbf{r}(\mathbf{C}, \mathbf{e}_i)$  is the  $i$ -th column of  $\mathbf{M}^{-1}$ . Since  $\mathbf{M}^{-1}$  exists (see Section 2.4.3), the vectors  $\mathbf{b}_i$  with  $i \in K$  have to be linearly independent. Thus,  $\mathbf{B}_K^\top \mathbf{B}_K$  is invertible.

**Remark:** Considering Equation (4.5), any reconstruction  $\mathbf{u} = \mathbf{r}(\mathbf{C}, \mathbf{g})$  is a vector in a  $|K|$ -dimensional subspace of  $\mathbb{R}^{|J|}$ , spanned by the inpainting echos  $\mathbf{b}_i$  with  $i \in K$ . Thereby, the basis formed by the inpainting echos is non-orthogonal: Since at least the  $i$ -th pixel is non-zero ( $i \in K$  and thus  $b_{ii} = e_{ii} = 1$ ) the inpainting echos are non-negative and thus the inner product  $\mathbf{b}_i^\top \mathbf{b}_j \neq 0$  for  $i \neq j$ .

Solving the minimisation problem (4.2) means essentially getting the vector which minimises the distance to  $\mathbf{f}$  in  $\mathbb{R}^{|J|}$ , which is basically nothing more than the projection of  $\mathbf{f}$  into the subspace.

### 4.3.2 Iterative Approach

The linear system given by the normal equations (4.7) can be solved exactly by using standard methods such as an LU-decomposition [Hig02]. However, since the system matrix is dense, this approach is rather slow. Therefore, we favour an iterative solver, namely the Gauss-Seidel method (cf. Equation (2.27)).

Let  $g_i^k$  denote the grey value of the  $i$ -th pixel in the  $k$ -th iteration, and let  $g_i^0 = f_i$  for all indices  $i$ . Then the corresponding Gauss-Seidel step for



solving the linear system of equations (4.7) with respect to  $\mathbf{g}_K$  is given as follows:

Consecutively for all  $i \in K$  do

$$g_i^{k+1} = \frac{1}{(\mathbf{B}_K^\top \mathbf{B}_K)_{i,i}} \cdot \left( (\mathbf{B}_K^\top \mathbf{f})_i - \sum_{\substack{j \in K \\ j < i}} (\mathbf{B}_K^\top \mathbf{B}_K)_{i,j} g_j^{k+1} - \sum_{\substack{j \in K \\ j > i}} (\mathbf{B}_K^\top \mathbf{B}_K)_{i,j} g_j^k \right) \quad (4.9)$$

$$= \frac{1}{\mathbf{b}_i^\top \mathbf{b}_i} \cdot \left( \mathbf{b}_i^\top \mathbf{f} - \sum_{\substack{j \in K \\ j < i}} \mathbf{b}_i^\top \mathbf{b}_j g_j^{k+1} - \sum_{\substack{j \in K \\ j > i}} \mathbf{b}_i^\top \mathbf{b}_j g_j^k \right) \quad (4.10)$$

$$= \frac{\mathbf{b}_i^\top \left( \mathbf{f} - \sum_{\substack{j \in K \\ j < i}} \mathbf{b}_j g_j^{k+1} - \sum_{\substack{j \in K \\ j \geq i}} \mathbf{b}_j g_j^k + \mathbf{b}_i g_i^k \right)}{\mathbf{b}_i^\top \mathbf{b}_i}. \quad (4.11)$$

Replacing  $\mathbf{b}_j$  in the sums by its definition and exploiting the linearity of  $\mathbf{r}$  yields:

$$g_i^{k+1} = \frac{\mathbf{b}_i^\top \left( \mathbf{f} - \sum_{\substack{j \in K \\ j < i}} \mathbf{r}(\mathbf{C}, \mathbf{e}_j) g_j^{k+1} - \sum_{\substack{j \in K \\ j \geq i}} \mathbf{r}(\mathbf{C}, \mathbf{e}_j) g_j^k + \mathbf{b}_i g_i^k \right)}{\mathbf{b}_i^\top \mathbf{b}_i} \quad (4.12)$$

$$= g_i^k + \frac{\mathbf{b}_i^\top \left( \mathbf{f} - \mathbf{r} \left( \mathbf{C}, \sum_{\substack{j \in K \\ j < i}} \mathbf{e}_j g_j^{k+1} + \sum_{\substack{j \in K \\ j \geq i}} \mathbf{e}_j g_j^k \right) \right)}{\mathbf{b}_i^\top \mathbf{b}_i}. \quad (4.13)$$

$$(4.14)$$

This result can further be simplified by defining the grey value vector  $\mathbf{g}$  to contain the “up-to-date” grey values at any step. That means right before updating the  $i$ -th entry of  $\mathbf{g}$ , it is given by

$$\mathbf{g} = \sum_{\substack{j \in K \\ j < i}} \mathbf{e}_j g_j^{k+1} + \sum_{\substack{j \in K \\ j \geq i}} \mathbf{e}_j g_j^k, \quad (4.15)$$

and when we do the update, the new  $\mathbf{g}$  is computed as

$$\mathbf{g}^{\text{new}} := \mathbf{g}^{\text{old}} + \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^{\text{old}}))}{\mathbf{b}_i^\top \mathbf{b}_i} \mathbf{e}_i. \quad (4.16)$$

Exploiting the linearity of  $\mathbf{r}$  we can also efficiently compute an “up-to-date” reconstruction  $\mathbf{u} = \mathbf{r}(\mathbf{C}, \mathbf{g})$  at each step:

$$\mathbf{u}^{\text{new}} = \mathbf{r}(\mathbf{C}, \mathbf{g}^{\text{new}}) \quad (4.17)$$

$$= \mathbf{r} \left( \mathbf{C}, \mathbf{g}^{\text{old}} + \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^{\text{old}}))}{\mathbf{b}_i^\top \mathbf{b}_i} \mathbf{e}_i \right) \quad (4.18)$$

$$= \mathbf{r}(\mathbf{C}, \mathbf{g}^{\text{old}}) + \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^{\text{old}}))}{\mathbf{b}_i^\top \mathbf{b}_i} \mathbf{r}(\mathbf{C}, \mathbf{e}_i) \quad (4.19)$$

$$= \mathbf{u}^{\text{old}} + \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{u}^{\text{old}})}{\mathbf{b}_i^\top \mathbf{b}_i} \mathbf{b}_i. \quad (4.20)$$

**Remark** Let us now for a moment consider the simplified optimisation problem, where we optimise only the  $i$ -th grey value  $g_i$  by computing how much the given one needs to be changed. Let  $\mathbf{g}$  be a given grey value vector and  $\mathbf{u} = \mathbf{r}(\mathbf{C}, \mathbf{g})$  its corresponding inpainting result. Then we aim for a solution of

$$\arg \min_{\gamma} |\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g} + \gamma \mathbf{e}_i)|^2. \quad (4.21)$$

Using the linearity of  $\mathbf{r}$  we get  $\mathbf{r}(\mathbf{C}, \mathbf{g} + \gamma \mathbf{e}_i) = \mathbf{u} + \gamma \mathbf{b}_i$ . We obtain the least squares solution

$$\gamma = \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{u})}{\mathbf{b}_i^\top \mathbf{b}_i}. \quad (4.22)$$

Thus, the new grey value vector is given by

$$\mathbf{g}^{\text{new}} = \mathbf{g}^{\text{old}} + \gamma \mathbf{e}_i. \quad (4.23)$$

This corresponds exactly to one Gauss-Seidel update step as given in Equation (4.16).

The complete algorithm is summarised as follows:

---

**Algorithm: Greyvalue optimisation**

---

**Input:**

Original image  $\mathbf{f}$ , inpainting mask  $\mathbf{C}$ .

**Initialisation:**

$\mathbf{u} := \mathbf{r}(\mathbf{C}, \mathbf{f})$  and  $\mathbf{g} := \mathbf{f}$ .

**Compute:**

For all  $i \in K$ :

    Compute the inpainting echo  $\mathbf{b}_i := \mathbf{r}(\mathbf{C}, \mathbf{e}_i)$ .

Do

    1. Set  $\mathbf{u}^{\text{old}} := \mathbf{u}$ .

    2. For all  $i \in K$ :

        a. Compute the correction term  $\gamma := \frac{\mathbf{b}_i^\top (\mathbf{f} - \mathbf{u})}{\mathbf{b}_i^\top \mathbf{b}_i}$ .

        b. Update the grey value  $g_i := g_i + \gamma$  and  
         the reconstruction  $\mathbf{u} := \mathbf{u} + \gamma \cdot \mathbf{b}_i$ .

    while  $|\text{MSE}(\mathbf{u}, \mathbf{f}) - \text{MSE}(\mathbf{u}^{\text{old}}, \mathbf{f})| > \varepsilon$ .

**Output:**

Optimised grey values  $\mathbf{g}$ .

---

Our algorithm terminates when the qualitative improvement from one to the next iteration step decreases to a value smaller than  $\varepsilon = 0.001$ . For the masks that are considered in this paper, we obtain the solution after not more than 10 Gauss-Seidel iterations.

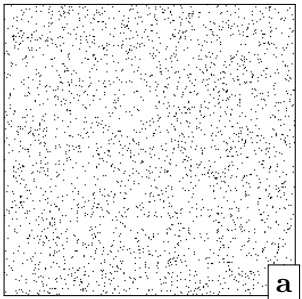
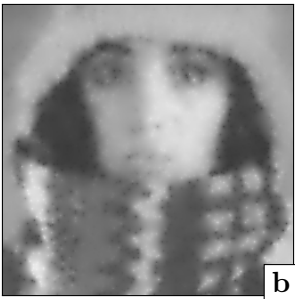
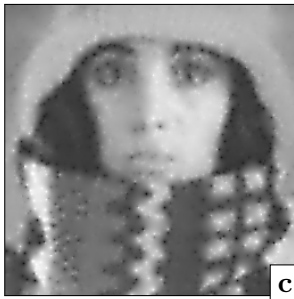
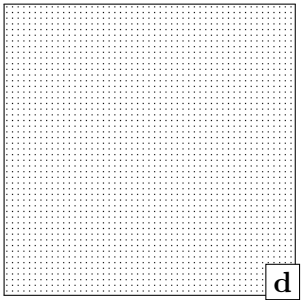


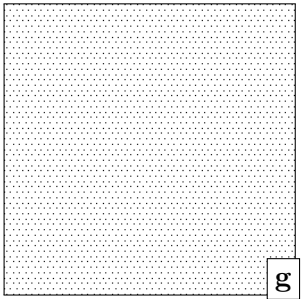
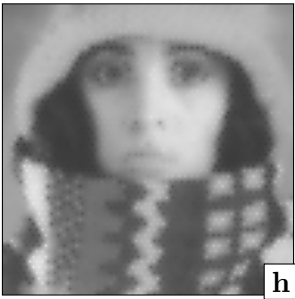
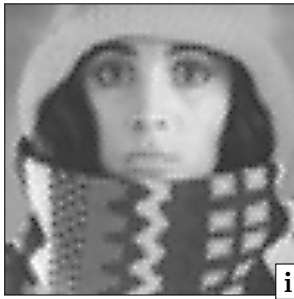
### 4.3.3 Results

In order to evaluate the capabilities of the grey value optimisation, we apply it to the all masks obtained for all test images so far (see first column of Figure 4.6 - Figure 4.11). The visual and quantitative results are depicted in the last column of Figure 4.6 - Figure 4.11. For reasons of clarity, Table 4.3 recapitulates the quantitative results. Indeed, we are able to improve the quantitative as well as the visual quality for all the reconstructions. Especially suboptimal masks benefit from this optimisation. This is because the

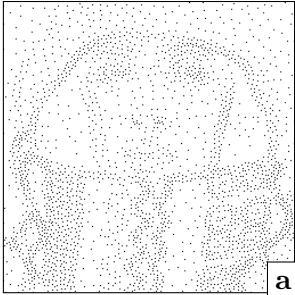
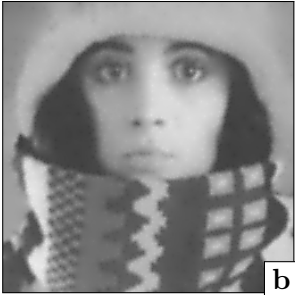
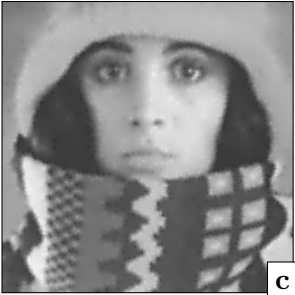
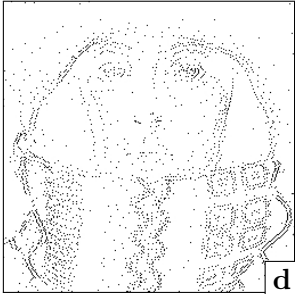
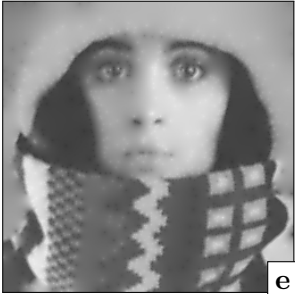


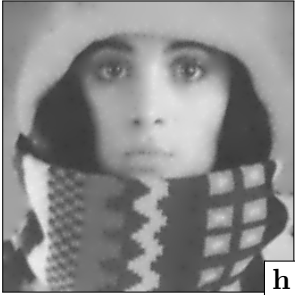
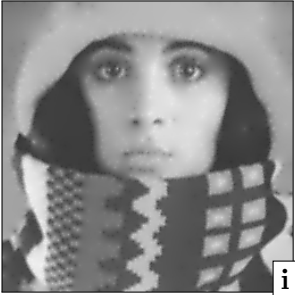
**Table 4.3:** Quantitative comparison of the reconstruction error (MSE) with 4% of all pixels for different test images and different inpainting data.

image	<i>trui</i>		<i>walter</i>		<i>peppers256</i>	
	no	yes	no	yes	no	yes
randomly selected	276.25	156.95	297.07	155.50	278.61	156.15
rectangular grid	181.72	101.62	184.00	91.97	185.04	104.41
hexagonal grid	187.08	102.74	181.82	89.97	180.22	101.62
analytic approach	84.04	42.00	39.85	20.19	70.05	43.77
	$(\sigma = 1.6, s = 0.80)$		$(\sigma = 1.5, s = 1.00)$		$(\sigma = 1.5, s = 0.95)$	
probabilistic sparsification	66.11	36.04	32.96	19.24	44.75	28.58
	$(p_{\text{cand}} = 0.3, p_{\text{rm}} = 0.000001)$		$(p_{\text{cand}} = 0.2, p_{\text{rm}} = 0.000001)$		$(p_{\text{cand}} = 0.1, p_{\text{rm}} = 0.000001)$	
non-local pixel exchange	41.92	<b>27.24</b>	18.37	<b>12.45</b>	29.63	<b>25.10</b>
	$(m = 20, 5 \cdot 10^5 \text{ iterations})$		$(m = 30, 5 \cdot 10^5 \text{ iterations})$		$(m = 30, 5 \cdot 10^5 \text{ iterations})$	

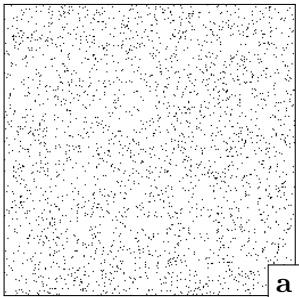


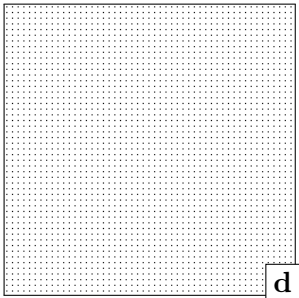


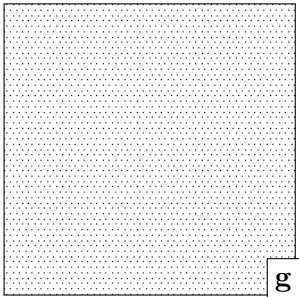


spatial optimisation approaches choose the optimal locations in dependence of the given grey values of the original image. Thus, the optimal spatial data fits to already given grey values. Greyvalue optimisation can compensate much more for the deficiencies that result of a suboptimal spatial selection. This becomes particularly apparent when considering the result for the random mask. It is remarkable how much the quality can be improved by simply choosing different grey values. This even holds for the spatially most optimal masks, namely the one given by the probabilistic sparsification in combination with the non-local pixel exchange. Compared to the random unoptimised mask, we are able to reduce the MSE by more than one order of magnitude for all test images. This shows the importance of data optimisation in PDE-based image compression.

	mask	reconstruction	reconstruction with optimal tonal data
randomly selected			
MSE		276.25	156.95
rectangular grid			
MSE		181.72	101.62
hexagonal grid			
MSE		187.08	102.74

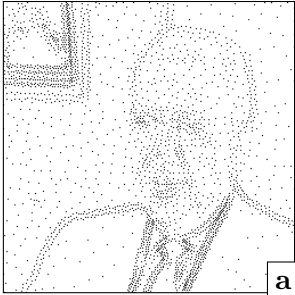








**Figure 4.6:** Evaluation of different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) random selection, (d) rectangular grid, (g) hexagonal grid. **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

	mask	reconstruction	reconstruction with optimal tonal data
analytic approach			
MSE		84.04	42.00
probabilistic sparsification			
MSE		66.11	36.04
+ non-local pixel exchange			
MSE		41.92	27.24

**Figure 4.7:** Evaluation of different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) analytic approach ( $s = 0.80$ ,  $\sigma = 1.6$ ), (d) probabilistic sparsification ( $p_{\text{cand}} = 0.3$ ,  $p_{\text{rm}} = 0.000001$ ), (g) non-local pixel exchange ( $m = 20$ , 500,000 iterations) applied to (d). **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

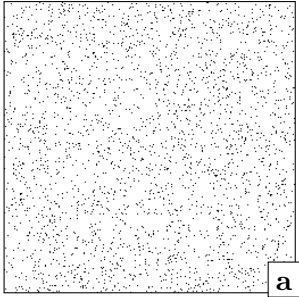
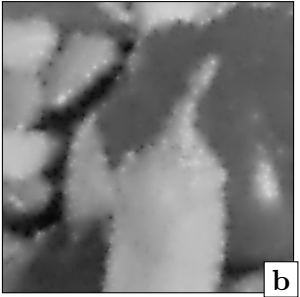
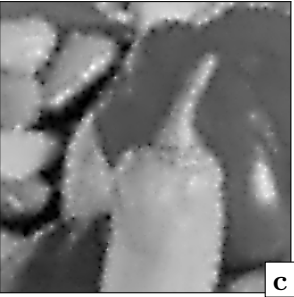
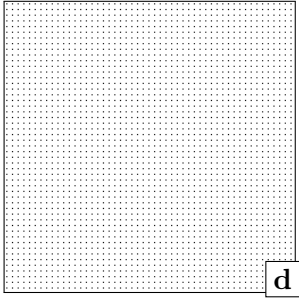
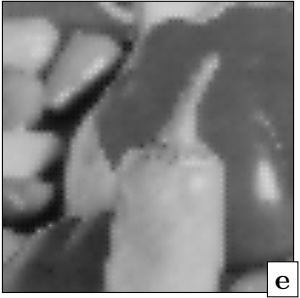
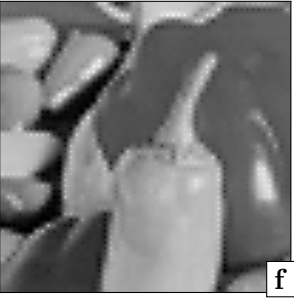
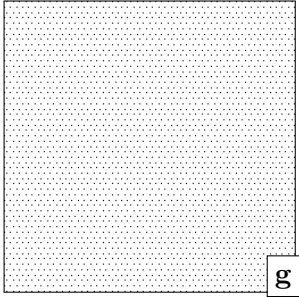
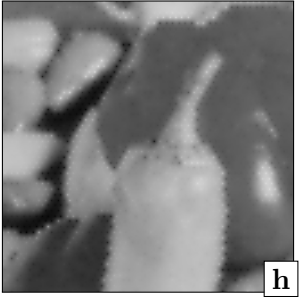

	mask	reconstruction	reconstruction with optimal tonal data
randomly selected			
MSE		297.07	155.50
rectangular grid			
MSE		184.00	91.97
hexagonal grid			
MSE		181.82	89.97

**Figure 4.8:** Evaluation of different inpainting data for the test image *walter* (see Figure 4.2(b)) using 4% of all pixels. **Left column:** Different masks obtained by (a) random selection, (d) rectangular grid, (g) hexagonal grid. **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

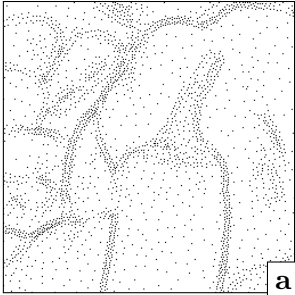
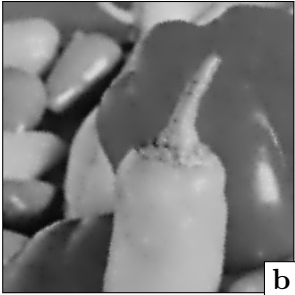
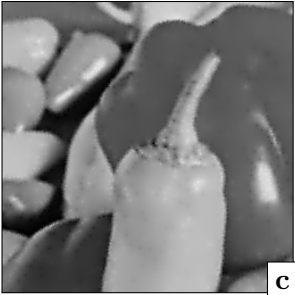


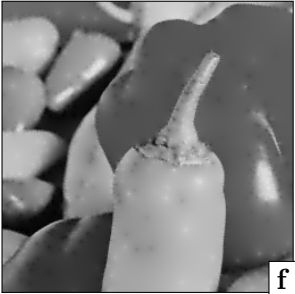
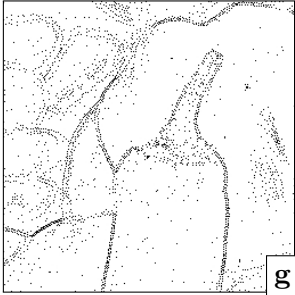


	mask	reconstruction	reconstruction with optimal tonal data
analytic approach			
MSE		39.85	20.19
probabilistic sparsification			
MSE		32.96	19.24
+ non-local pixel exchange			
MSE		18.37	12.45

**Figure 4.9:** Evaluation of different inpainting data for the test image *walter* (see Figure 4.2(b)) using 4% of all pixels. **Left column:** Different masks obtained by (a) analytic approach ( $s = 1.00$ ,  $\sigma = 1.5$ ), (d) probabilistic sparsification ( $p_{\text{cand}} = 0.2$ ,  $p_{\text{rm}} = 0.000001$ ), (g) non-local pixel exchange ( $m = 30$ , 500,000 iterations) applied to (d). **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.



	mask	reconstruction	reconstruction with optimal tonal data
randomly selected			
MSE		278.61	156.15
rectangular grid			
MSE		185.04	104.41
hexagonal grid			
MSE		180.22	101.62

**Figure 4.10:** Evaluation of different inpainting data for the test image *peppers256* (see Figure 4.2(c)) using 4% of all pixels. **Left column:** Different masks obtained by (a) random selection, (d) rectangular grid, (g) hexagonal grid. **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

	mask	reconstruction	reconstruction with optimal tonal data
analytic approach			
MSE		70.05	43.77
probabilistic sparsification			
MSE		44.75	28.58
+ non-local pixel exchange			
MSE		29.63	25.1

**Figure 4.11:** Evaluation of different inpainting data for the test image *peppers256* (see Figure 4.2(c)) using 4% of all pixels. **Left column:** Different masks obtained by (a) analytic approach ( $s = 0.95$ ,  $\sigma = 1.5$ ), (d) probabilistic sparsification ( $p_{\text{cand}} = 0.1$ ,  $p_{\text{rm}} = 0.000001$ ), (g) non-local pixel exchange ( $m = 30$ , 500,000 iterations) applied to (d). **Middle column:** Reconstructions with homogeneous diffusion inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

## 4.4 Extensions to the Biharmonic Operator and Edge-Enhancing Diffusion

In the previous section, we saw that a careful optimisation of the interpolation data allows us to obtain impressive reconstructions with only 4% of all pixels. Those results get even more impressive in view of the fact that we used an interpolation operator which usually gives only poor reconstructions. In this section we want to answer the question to what extent the spatial and tonal optimisation algorithms of the previous sections can be used with, or extended to more advanced inpainting operators. To this end, we will consider the two operators introduced in Section 2.6, namely the *biharmonic* operator and the one given by *edge enhancing-diffusion (EED)*.

### 4.4.1 Spatial Optimisation

Let us first consider the spatial optimisation. The theory of the analytic approach bases on homogeneous diffusion and cannot easily be extended to more sophisticated inpainting operators. Therefore, we will not consider this approach in this section. In contrast, the probabilistic approach as well as the non-local pixel exchange can be used without any restrictions or changes. As before, we search for the best parameters of the optimisation methods. The two EED parameters are kept fixed as  $\lambda = 0.8$  and  $\sigma = 0.7$ . Experiments have shown that this is a reasonable choice, giving results of high quality.

### Results with the Biharmonic Operator

The resulting masks and reconstructions with the biharmonic operator when applied to the test image *trui* are shown in the left and middle column of Figure 4.13. For comparison we also applied the inpainting method to the randomly selected as well as the rectangular and hexagonal grid masks (see Figure 4.12, left and middle column). Table 4.4 allows an easy quantitative comparison of the obtained results.

For the unoptimised masks the biharmonic operator in general performs better than the homogeneous one. The reconstructions clearly benefit from the additional smoothness at the given inpainting data. This does not hold for probabilistic sparsification. There, the biharmonic operator creates strong over- and undershoots which outweigh the benefits of additional smoothness. Indeed, the minimum and maximum values of the given reconstruction lie around  $-59$  and  $346$ , respectively. This is far beyond the range of the original image with minimum and maximum values  $56$  and  $241$ .

The reason for this behaviour is that at the beginning probabilistic sparsification mainly removes pixels in homogeneous areas as those can easily be reconstructed. Edges at that stage are still finely represented, as there is still enough inpainting data located. Therefore, for a larger number of mask pixels, the biharmonic operator can usually outperform the homogeneous one when using probabilistic sparsification. However, for very sparse masks, such as in our case, the algorithm will also remove some important pixels at edges in particular during later iterations. As a result, the biharmonic operator creates over- and undershoots in the reconstruction. In general it holds that the less data is given and the more non-uniformly this data is distributed, the more likely it is that the biharmonic operator creates such over- and undershoots.

With additional non-local pixel exchange we can overcome this problem. It redistributes the mask pixels to more favourable locations. A comparison of the mask shows that indeed several pixels have been relocated into homogeneous areas in order to prevent the over- and undershoots. The minimal and maximal grey values return to 47 and 248, respectively. As a consequence, the MSE falls from 79.26 to 20.89, which is again below the one of homogeneous diffusion inpainting.

### Results with EED

The results for EED are analogously given in Figure 4.14 and Figure 4.15, left and middle column. Table 4.4 summarises the quantitative results.

Compared to the biharmonic operator, the reconstructions of the unoptimised masks can merely benefit from the good reconstruction capabilities of EED. The corresponding MSE is only slightly smaller. In contrast, in the case of probabilistic sparsification EED is clearly superior. There, EED does not create over and undershoots, but still gives smooth reconstructions, and allows even to recreate edges. We obtain an MSE of 24.20 which is already lower than the overall best result obtained with homogeneous diffusion. By applying additionally the non-local pixel exchange we even reach an error of only 12.62.

Figure 4.16, left column allows a direct comparison of the best masks we could find for each of the inpainting operators. Especially when comparing the best mask of homogeneous diffusion inpainting with the others, we can observe that optimal masks from different operators can differ quite a lot. Even though there is no obvious structural difference between the biharmonic mask and the EED mask the differences become evident if we exchange the corresponding masks: Reconstructing with EED using the biharmonic-mask gives an error of 51.17 instead of 12.62. Even more signi-

ificant are the differences if the biharmonic operator is used in combination with the optimal EED-mask. There we obtain an MSE of 222.27 instead of 20.89. Thus, the selected optimal data highly depends on the chosen operator.

#### 4.4.2 Greyvalue Optimisation

Let us now see how we can optimise the grey values for the obtained masks. In Section 4.3 we optimised the grey values by considering the minimisation approach (4.2) and by making use of the fact that the inpainting function  $\mathbf{r}$  is linear in its second argument, i.e. the given grey values. As it is for the Laplace operator, the biharmonic operator  $\Delta^2$  is linear. Therefore, our approach for grey value optimisation can be transferred one-to-one. Unfortunately, this does not hold for EED. There the inpainting operator, represented in the matrix  $\mathbf{A}$ , now depends on  $\mathbf{u}$  (see Equation (2.57)). Its corresponding inpainting function  $\mathbf{r}$  is in this case no longer linear in its second argument and Equation (4.2) becomes a non-linear least squares problem. Before we discuss how to solve it, let us first have a look at the results for the biharmonic operator when using the algorithm as introduced in Section 4.3.2.

#### Results with Biharmonic Operator

The results for the grey value optimisation with the biharmonic operator are given in the right column of Figure 4.12 and Figure 4.13. As before, the quantitative results can also be found in Table 4.4. Especially in the case of probabilistic sparsification, the gain in quality is impressive: The MSE can be reduced to 23.50. This is due to the fact that by adapting the grey values, over- and undershoots are damped. Indeed, after optimisation the values lie between 32 and 254 instead of  $-59$  and 346, respectively. However, for the best mask, namely the one given after the non-local pixel exchange, the improvements are not that tremendous since the over- and undershoots are anyway not as predominant as in the case of probabilistic sparsification. Still, we can improve the result such that we end up with an MSE of 16.73.

#### Greyvalue Optimisation in the Non-Linear Case

Let us now come back to the grey value optimisation in the case of EED. As we mentioned before, Equation (4.2) becomes a non-linear least squares problem. In order to solve it, we will apply the Gauss-Newton

algorithm [Bj96]: Starting with  $\mathbf{g}^0 = \mathbf{f}$  the optimal grey values are approximated by the iterative scheme

$$\mathbf{g}^{s+1} = \mathbf{g}^s + \underbrace{(\mathbf{J}_r^{s\top} \mathbf{J}_r^s)^{-1} \mathbf{J}_r^{s\top} (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^s))}_{=: \boldsymbol{\alpha}^s}, \quad (4.24)$$

where  $(\mathbf{J}_r^s)_{i,j} = (\mathbf{J}_r(\mathbf{C}, \mathbf{g}^s))_{i,j} = \frac{\partial r_i}{\partial f_j}(\mathbf{C}, \mathbf{g}^s)$  is the Jacobian matrix of  $\mathbf{r}$  at  $\mathbf{g}^s$ . One determines  $\boldsymbol{\alpha}^s$  by solving the linear system of equations

$$\mathbf{J}_r^{s\top} \mathbf{J}_r^s \boldsymbol{\alpha}^s = \mathbf{J}_r^{s\top} (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^s)). \quad (4.25)$$

The entries of the Jacobian matrix are approximated by finite differences:

$$(\mathbf{J}_r(\mathbf{C}, \mathbf{f}))_{i,j} \approx \frac{r_i(\mathbf{C}, \mathbf{f} + \delta \mathbf{e}_j) - r_i(\mathbf{C}, \mathbf{f})}{\delta}, \quad (4.26)$$

where  $\delta$  should usually be a small constant. Moreover, we denote the  $j$ -th column vector of  $\mathbf{J}_r(\mathbf{C}, \mathbf{f})$  by  $\mathbf{r}_{f_j}(\mathbf{C}, \mathbf{f})$  and call it the *inpainting derivative* of  $\mathbf{r}$  for the  $j$ -th pixel:

$$\mathbf{r}_{f_j}(\mathbf{C}, \mathbf{f}) \approx \frac{\mathbf{r}(\mathbf{C}, \mathbf{f} + \delta \mathbf{e}_j) - \mathbf{r}(\mathbf{C}, \mathbf{f})}{\delta}. \quad (4.27)$$

Note that  $(\mathbf{J}_r(\mathbf{C}, \mathbf{f}))_{i,j} = 0$  if  $j \in J \setminus K$ . Thus the inpainting derivative for the  $j$ -th pixel vanishes if  $j \in J \setminus K$ , similar as in the case of the inpainting echo (see Equation (4.4)). Indeed, it is easy to verify that inpainting derivative and inpainting echo are equal in the linear case. For the equation system (4.25) this means that  $\alpha_j$  can be chosen arbitrarily if  $j \in J \setminus K$ . The remaining  $\alpha_i$  with  $i \in K$  can be obtained by considering the simplified equation system (cf. Equation (4.6))

$$\mathbf{D}_K^{s\top} \mathbf{D}_K^s \boldsymbol{\alpha}_K^s = \mathbf{D}_K^{s\top} (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^s)). \quad (4.28)$$

where  $\boldsymbol{\alpha}_K^s = (\alpha_i^s)_{i \in K}$  is a vector of size  $|K|$ , and  $\mathbf{D}_K^s$  is a  $|J| \times |K|$  matrix that contains the vectors  $\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)$ ,  $i \in K$ , as columns. Similar to the system (4.7) we can solve this linear system of equations by using the Gauss-Seidel method. We denote the  $i$ -th value of  $\boldsymbol{\alpha}^s$  in the  $k$ -th iteration by  $\alpha_i^{s,k}$  and initialise with 0, i.e.,  $\alpha_i^{s,0} := 0$  for all indices  $i$ . The corresponding Gauss-

Seidel iteration step is then given by (cf. Equations 4.9-4.12))

$$\alpha_i^{s,k+1} = \frac{1}{(\mathbf{D}_K^{s\top} \mathbf{D}_K^s)_{i,i}} \cdot \left( (\mathbf{D}_K^{s\top} (\mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^s)))_i - \sum_{\substack{j \in K \\ j < i}} (\mathbf{D}_K^{s\top} \mathbf{D}_K^s)_{i,j} \alpha_j^{s,k+1} - \sum_{\substack{j \in K \\ j > i}} (\mathbf{D}_K^{s\top} \mathbf{D}_K^s)_{i,j} \alpha_j^{s,k} \right) \quad (4.29)$$

$$= \alpha_i^{s,k} + \frac{\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)^\top}{|\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)|^2} \cdot \left( \mathbf{f} - \mathbf{r}(\mathbf{C}, \mathbf{g}^s) - \sum_{\substack{j \in K \\ j < i}} \mathbf{r}_{f_j}(\mathbf{C}, \mathbf{g}^s) \alpha_j^{s,k+1} - \sum_{\substack{j \in K \\ j \geq i}} \mathbf{r}_{f_j}(\mathbf{C}, \mathbf{g}^s) \alpha_j^{s,k} \right). \quad (4.30)$$

To simplify this equation further, we define in each Gauss-Newton step a vector  $\boldsymbol{\beta}^s$  which is recomputed whenever a new  $\alpha_i^{s,k+1}$  is available:

$$\boldsymbol{\beta}^s := \mathbf{r}(\mathbf{C}, \mathbf{g}^s) + \sum_{\substack{j \in K \\ j < i}} \mathbf{r}_{f_j}(\mathbf{C}, \mathbf{g}^s) \alpha_j^{s,k+1} + \sum_{\substack{j \in K \\ j \geq i}} \mathbf{r}_{f_j}(\mathbf{C}, \mathbf{g}^s) \alpha_j^{s,k}. \quad (4.31)$$

Note that  $\boldsymbol{\beta}^s$  can be considered to be a linearised version of the reconstruction using the most up-to-date grey values based on the latest  $\mathbf{g}^s$  and the currently newest values of  $\boldsymbol{\alpha}$ :

$$\mathbf{r} \left( \mathbf{C}, \mathbf{g}^s + \sum_{\substack{j \in K \\ j < i}} \mathbf{e}_j \alpha_j^{s,k+1} + \sum_{\substack{j \in K \\ j \geq i}} \mathbf{e}_j \alpha_j^{s,k} \right). \quad (4.32)$$

With this vector, Equation (4.30) becomes

$$\alpha_i^{s,k+1} = \alpha_i^{s,k} + \frac{\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)^\top (\mathbf{f} - \boldsymbol{\beta}^s)}{|\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)|^2}. \quad (4.33)$$

Instead of computing  $\boldsymbol{\beta}^s$  everytime completely, it can also be obtained by updating its previous value using the old and the newly obtained  $\alpha$ -value, i.e.  $\alpha_i^{s,k}$  and  $\alpha_i^{s,k+1}$ , respectively:

$$\boldsymbol{\beta}^{s;\text{new}} := \boldsymbol{\beta}^{s;\text{old}} + \alpha_i^{s,k+1} \cdot \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s) - \alpha_i^{s,k} \cdot \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s) \quad (4.34)$$

$$= \boldsymbol{\beta}^{s;\text{old}} + (\alpha_i^{s,k+1} - \alpha_i^{s,k}) \cdot \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s), \quad (4.35)$$

where  $\boldsymbol{\beta}^{\text{old}}$  is initially set to  $\mathbf{r}(\mathbf{C}, \mathbf{g}^s)$  since  $\boldsymbol{\alpha}^{s,0} = \mathbf{0}$ . Moreover, by bringing  $\alpha_i^{s,k}$  in Equation (4.33) to the left-hand-side, we can replace  $(\alpha_i^{s,k+1} - \alpha_i^{s,k})$  in Equation (4.35) and obtain (cf. Equation (4.20)):

$$\boldsymbol{\beta}^{s;\text{new}} := \boldsymbol{\beta}^{s;\text{old}} + \frac{\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)^\top (\mathbf{f} - \boldsymbol{\beta}^{s;\text{old}})}{|\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)|^2} \cdot \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s). \quad (4.36)$$

Eventually, we obtain a solution  $\boldsymbol{\alpha}^s$  which can be used to perform the Gauss-Newton step given in Equation (4.24). However, instead of computing  $\boldsymbol{\alpha}^s$  explicitly, and adding it to  $\mathbf{g}^s$  we can also simply update the grey value vector  $\mathbf{g}$  immediately whenever a new value  $\alpha_i^{s,k+1}$  is obtained. That means we simply subtract the old  $\alpha$ -value and add the new one in the corresponding pixel  $i$ :

$$\mathbf{g}^{\text{new}} := \mathbf{g}^{\text{old}} - \alpha_i^{s,k} \cdot \mathbf{e}_i + \alpha_i^{s,k+1} \cdot \mathbf{e}_i \quad (4.37)$$

$$= \mathbf{g}^{\text{old}} + (\alpha_i^{s,k+1} - \alpha_i^{s,k}) \cdot \mathbf{e}_i. \quad (4.38)$$

In the same way how we did it for  $\boldsymbol{\beta}^s$  (see Equations (4.35) and (4.36)), we can then replace  $(\alpha_i^{s,k+1} - \alpha_i^{s,k})$ . This yields (cf. Equation (4.16))

$$\mathbf{g}^{\text{new}} := \mathbf{g}^{\text{old}} + \frac{\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)^\top (\mathbf{f} - \boldsymbol{\beta}^{s;\text{old}})}{|\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)|^2} \cdot \mathbf{e}_i. \quad (4.39)$$

Regarding the inpainting derivative  $\mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}^s)$ , it is important to differentiate between the up-to-date grey value vector  $\mathbf{g}$  and the vector  $\mathbf{g}^s$ . Whereas  $\mathbf{g}$  is already updated during the Gauss-Seidel iterations,  $\mathbf{g}^s$  contains the grey values as they are given in the  $s$ -th Gauss-Newton iteration right before the first Gauss-Seidel step is even performed.

Unfortunately, with the Gauss-Newton method the sum of squares may not decrease at every iteration. Indeed, it turns out that our nonlinear least squares problem is highly ill-posed such that the method easily starts oscillating around the steady state. A common remedy to this problem is to introduce a damping parameter  $\omega \in [0, 1]$  which is multiplied with  $\boldsymbol{\alpha}^s$  in every step. As a result, the oscillations are reduced at the cost of fast convergence. Since there is still no guarantee of convergence, the MSE between the current and the previous reconstruction, as done in the linear case, is not a reliable stopping criterion. Instead, we specify the number of Gauss-Newton and Gauss-Seidel iterations manually. One might argue that in the case of the Gauss-Seidel iterations, we should not limit the number of iterations, but instead solve the underlying linear system of equations (4.28) as exactly as possible. However, its system matrix depends on the inpainting derivatives, which in turn depend on the the optimised grey



values that we aim to compute. Thus, it can even be beneficial not to solve this system of equations exactly such that the nonlinear terms, i.e. the inpainting derivatives are updated more frequently.

In addition, we do not return the very last result at the end of the algorithm but instead the best one we obtained during the process up to this point. Summarising, the grey value optimisation in the non-linear case is given by the following algorithm:

---

**Algorithm: Greyvalue optimisation (nonlinear)**

---

**Input:**

Original image  $\mathbf{f}$ , inpainting mask  $\mathbf{C}$ , damping factor  $\omega$ , parameter  $\delta$  to approximate inpainting derivative, maximal number of Gauss-Seidel iterations  $n_{\text{GS}}$ , maximal number of Gauss-Newton iterations  $n_{\text{GN}}$ .

**Initialisation:**

$\mathbf{g} := \mathbf{f}$ ,  $\mathbf{g}^{\text{best}} := \mathbf{f}$  and  $\mathbf{u}^{\text{best}} := \mathbf{r}(\mathbf{C}, \mathbf{f})$ .

**Compute:**

For  $n_{\text{GN}}$  iterations do

1. Initialise  $\boldsymbol{\beta} := \mathbf{r}(\mathbf{C}, \mathbf{g})$ .
2. If  $\text{MSE}(\boldsymbol{\beta}, \mathbf{f}) < \text{MSE}(\mathbf{u}^{\text{best}}, \mathbf{f})$   
 $\mathbf{u}^{\text{best}} := \boldsymbol{\beta}$  and  $\mathbf{g}^{\text{best}} := \mathbf{g}$
3. For all  $i \in K$ :  
 Compute the inpainting derivatives

$$\mathbf{d}_i := \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{g}) \approx \frac{\mathbf{r}(\mathbf{C}, \mathbf{g} + \delta \mathbf{e}_i) - \boldsymbol{\beta}}{\delta}.$$

4. For  $n_{\text{GS}}$  iterations do

For all  $i \in K$ :

- a. Compute the correction term  $\gamma := \frac{\mathbf{d}_i^\top (\mathbf{f} - \boldsymbol{\beta})}{|\mathbf{d}_i|^2}$ .
- b. Update the grey value  $g_i := g_i + \omega \gamma$   
 and  $\boldsymbol{\beta} := \boldsymbol{\beta} + \omega \gamma \cdot \mathbf{d}_i$ .

**Output:**

Optimised grey values  $\mathbf{g}^{\text{best}}$ .

---

### Results with EED

In total our algorithm offers four degrees of freedom: the number of outer Gauss-Newton iterations  $n_{\text{GN}}$ , the number of inner Gauss-Seidel iterations  $n_{\text{GS}}$ , the derivative parameter  $\delta$  and the damping parameter  $\omega$ . We expect more stable solutions by choosing the damping parameter  $\omega$  to be small. To approximate the inpainting derivative accurately, the derivative parameter  $\delta$  should be small as well. A reasonable choice for the number of Gauss-Seidel iterations  $n_{\text{GS}}$  would be to choose as many as necessary to solve the underlying linear system with a sufficient accuracy. The number of of Gauss-Newton iterations should be chosen large enough to ensure convergence.

However, such a parameter choice is not feasible in terms of run time: The smaller  $\omega$ , the more Gauss-Newton iterations  $n_{\text{GN}}$  are necessary to ensure convergence. Moreover, with very small  $\omega$  the method can easily get trapped in local minimal. In addition, due to numerical issues a small value for  $\delta$  can lead to unstable results, such that the method might even not decrease, but instead increase the error. Last but not least, as already mentioned earlier, it can be advantageous to choose the number of Gauss-Seidel iterations  $n_{\text{GS}}$  small such that the nonlinear terms are updated more frequently.

To get a more determined feeling about suitable parameters, we run several experiments using different settings. We analyse the results given after only 20 and then after 1000 Gauss-Newton iterations to see the algorithm's behaviour at an early and an later stage. Even on modern architectures, the runtime for the latter on some parameter combinations lies already in the order of weeks. This is why we refrain from testing higher numbers. For the number of Gauss-Seidel iterations  $n_{\text{GS}}$ , we tested the values 1, 10, 20, 100, for the derivative parameter  $\delta$  the values 0.01, 0.1, 1, 10, 100, and for the damping parameter  $\omega$  the values 0.001, 0.01, 0.1 and 1.

Tables 4.5 and 4.8 show the best results for all our masks with respect to each considered value of the number of Gauss-Seidel iterations. Tables 4.6 and 4.9 and Tables 4.7 and 4.10 state the analogous results for the derivative parameter  $\delta$  and the damping parameter  $\omega$ .

A first general observation is that with the right parameters, only 20 Gauss-Newton iterations suffice to already obtain results that are almost as good as the ones obtained after 1000 iterations. We also see that at both stages, the other optimal parameters differ slightly.

A closer look at the tables for the Gauss-Seidel iterations and the damping parameter shows that after 1000 Gauss-Newton iterations there is no clear tendency to favour specific parameter values over others. For any  $\omega$

and any number of Gauss-Seidel iterations we can find a suitable setting and get results of similar quality.

However, there seems to be a dependency between both parameters: The higher the  $\omega$  the lower  $n_{GS}$  should be chosen and vice versa. This holds all the more at the early stages of the algorithm, as we can see from the results after 20 Gauss-Newton iterations. This is mainly due to the fact that both parameters influence the speed of convergence.

At the early stages of the algorithm we can also observe that the appropriate choice of  $\omega$  matters more. There it is advisable not to choose it too small, because such values slow the convergence strongly down. A damping parameter of 0.01 again is already fine. Choosing  $\omega = 0.1$  usually gives the best results at such an early stage. Also with  $\omega = 1$  good results can be obtained. However, especially for spatially suboptimal data as in the case of the random mask,  $\omega = 0.1$  is still preferable. This is even the case when only one Gauss-Newton iteration is performed. At later stages (see  $n_{GN} = 1000$ ) smaller values like 0.01 or 0.1 should be favoured. They restrict the latitude of the swings and can thus lead to better results. For higher numbers of Gauss-Newton iterations even a stronger damping might be considered.

For the number of Gauss-Seidel iterations even at early stages of the algorithms the quality of the shown results is quite similar. On one hand this shows that solving the linear system exactly is not necessary, or that frequent updates of the non-linearities can be neglected, provided the remaining parameters especially  $\omega$ , are appropriately adapted. On the other hand, it means that since the choice of omega matters at early stages, we should adapt the Gauss-Seidel iterations in dependence of  $\omega$  and not vice versa. In combination with the suggested values for  $\omega$ , i.e.  $\omega = 0.1$  or  $\omega = 0.01$ , usually  $n_{GS} = 30$  gives good results.

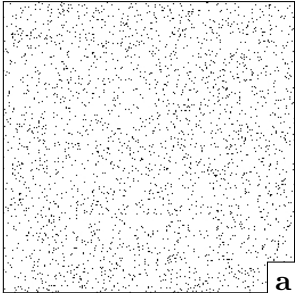


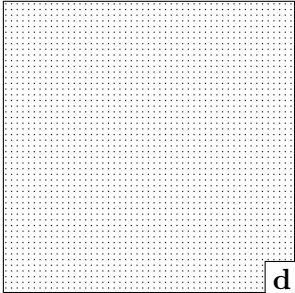


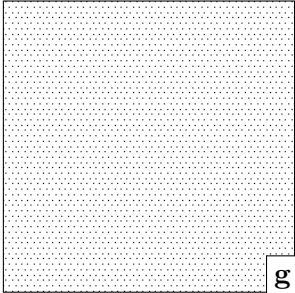
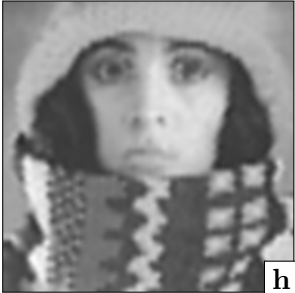
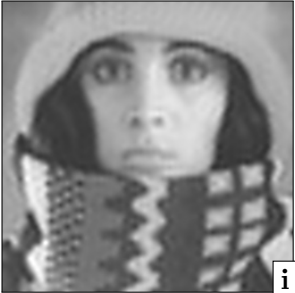
In contrast to  $\omega$  and  $n_{GS}$ , the derivative parameter  $\delta$  shows a clear tendency: It should neither be chosen too large nor too small. At early stages,  $\delta = 10$  usually gives the best results, whereas at later stages most often  $\delta = 1$  leads to better results. Even for much higher numbers of Gauss-Newton iterations, the latter value seems to be an appropriate choice. Note that these values are unexpected high considering the fact that the grey values in an image usually range from 0 to 255. However, with smaller values the method often already gets stuck at early stages of the algorithm. This behaviour can be attributed to the EED-inpainting operator: When the grey value of a pixel is changed within a specific range the reconstruction result does not change, or changes only marginally. At least this is the case with a common inpainting parameter setting, as the one we are using here. Then suddenly, when a certain value is exceeded, the changes in

the reconstruction are extensive, since the grey value change can cause the creation of new edges. For our algorithm this means that if  $\delta$  is too small, the inpainting derivative is close to 0, and therefore the current grey value is already considered to be optimal. In contrast, with too large values for  $\delta$ , for instance  $\delta = 100$ , the estimation of the inpainting derivative becomes too inaccurate. As a consequence the found optimal grey values will be inaccurate as well, and thus the quality of the reconstructions is suboptimal.

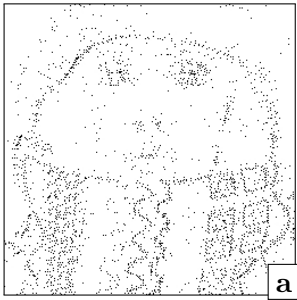

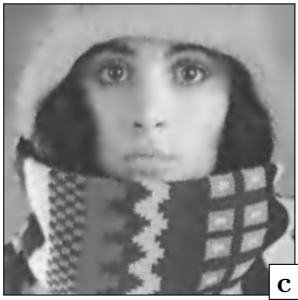
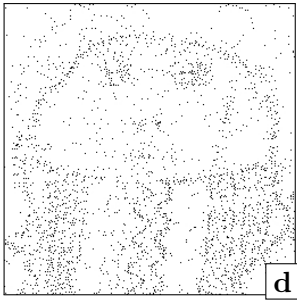


Summarised, a reasonable combination of parameters seems to be  $n_{GS} = 30$ ,  $\delta = 10$  and  $\omega = 0.1$ . With this setting one can obtain results of good quality already after only a few Gauss-Newton iterations. Table 4.4 allows a comparison to the reconstructions obtained with unoptimised grey values as well as the ones obtained with the other operators. Here, we choose the parameters with which we obtained the best results after 1000 Gauss-Newton iterations (see bold printed lines in the Tables 4.8, 4.9, and 4.10). As we can see, the results outperform the previous once for any mask. This also holds for the visual comparison given in the Figures 4.14 and 4.15 as well as Figure 4.16. The overall best result was obtained with the mask of the non-local pixel exchange and depicts an MSE of 10.84. The quality of the corresponding reconstruction is quiet impressive, especially since it was obtained by only 4% of all pixels. Even in the zoom-in depicted in Figure 4.16(k) one can identify only minor differences compared to the original.

**Table 4.4:** Quantitative comparison of the reconstruction error (MSE) with 4% of all pixels for different inpainting operators and different inpainting data. The parameters for the grey value optimisation in the case of EED have been chosen as given by the bold printed lines in Table 4.10. Note that in the case of EED the tonal data has been optimised but is not necessarily optimal.

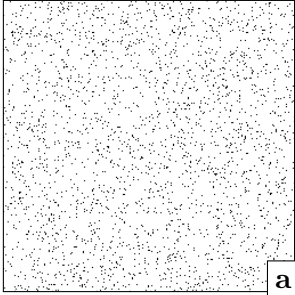
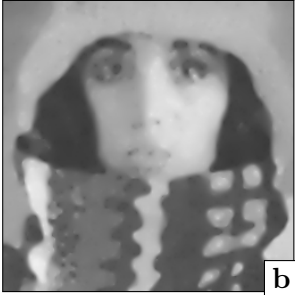
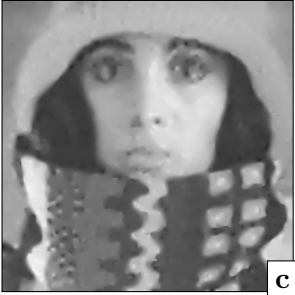
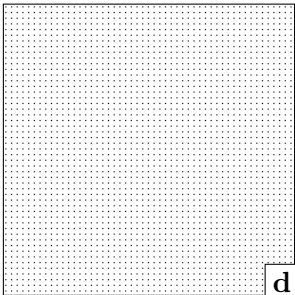
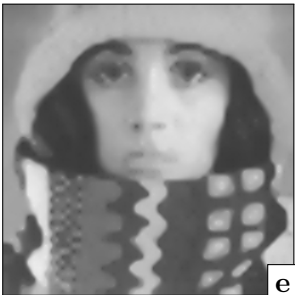

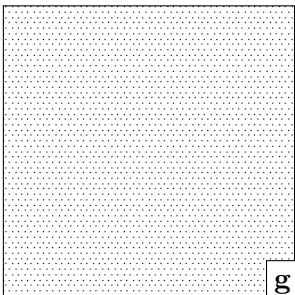
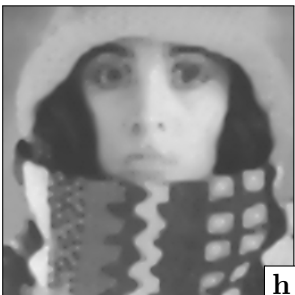

operator	<i>homogeneous</i>		<i>biharmonic</i>		<i>EED</i> ( $\lambda = 0.8, \sigma = 0.7$ )	
	no	yes	no	yes	no	(yes)
randomly selected	276.25	156.95	193.86	93.49	181.81	90.83
rectangular grid	181.72	101.62	107.96	74.01	102.85	62.34
hexagonal grid	187.08	102.74	109.01	72.06	102.83	61.21
probabilistic sparsification	66.11 ( $p_{\text{cand}} = 0.3$ , $p_{\text{rm}} = 0.000001$ )	36.04	79.26 ( $p_{\text{cand}} = 0.005$ , $p_{\text{rm}} = 0.000001$ )	23.50	24.20 ( $p_{\text{cand}} = 0.05$ , $p_{\text{rm}} = 0.000001$ )	14.53
non-local pixel exchange	41.92 ( $m = 20$ , $5 \cdot 10^5$ iterations)	<b>27.24</b>	20.89 ( $m = 10$ , $5 \cdot 10^5$ iterations)	<b>16.73</b>	12.62 ( $m = 30$ , $5 \cdot 10^5$ iterations)	<b>10.84</b>

	mask	reconstruction	reconstruction with optimal tonal data
randomly selected			
MSE		193.86	93.49
rectangular grid			
MSE		107.96	74.01
hexagonal grid			
MSE		109.01	72.06

**Figure 4.12:** Evaluation of the biharmonic operator with different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) random selection, (d) rectangular grid, (g) hexagonal grid. **Middle column:** Reconstructions with biharmonic inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.

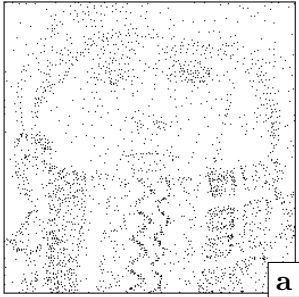

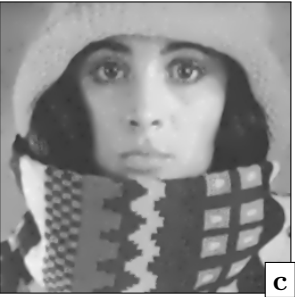
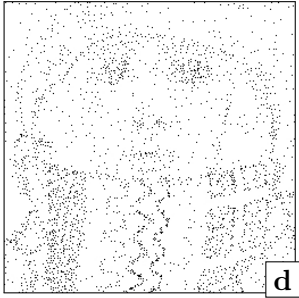


	mask	reconstruction	reconstruction with optimal tonal data
probabilistic sparsification			
MSE		79.26	23.50
+ non-local pixel exchange			
MSE		20.89	16.73

**Figure 4.13:** Evaluation of the biharmonic operator with different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) probabilistic sparsification ( $p_{\text{cand}} = 0.005$ ,  $p_{\text{rm}} = 0.000001$ ), (d) non-local pixel exchange ( $m = 10$ , 500,000 iterations) applied to (a). **Middle column:** Reconstructions with biharmonic inpainting using the masks (a,d). **Right column:** Similar to middle column, but using optimal tonal data.

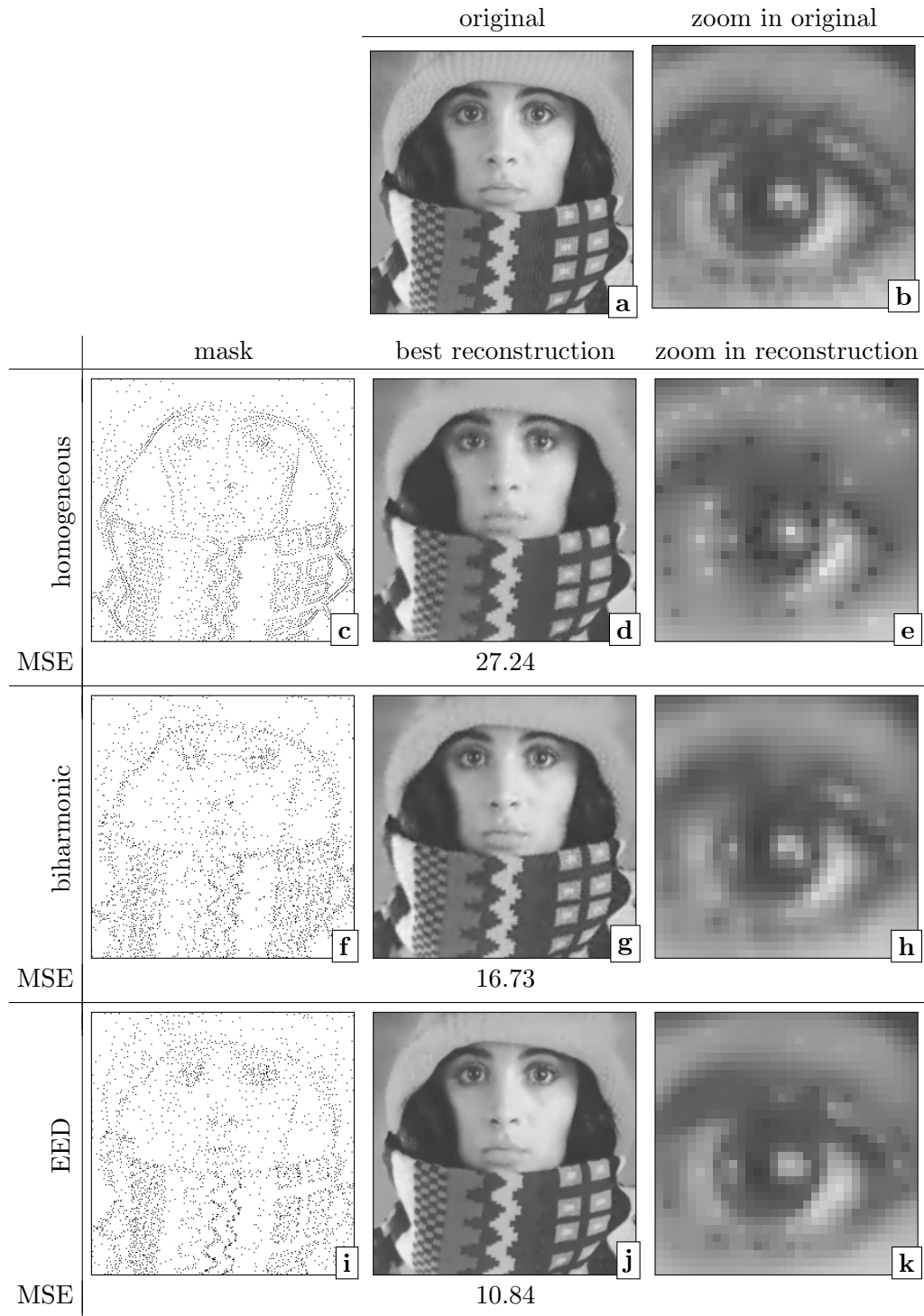
	mask	reconstruction	reconstruction with optimal tonal data
randomly selected			
MSE		181.81	90.83
rectangular grid			
MSE		102.85	62.34
hexagonal grid			
MSE		102.83	61.21

**Figure 4.14:** Evaluation of EED with different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) random selection, (d) rectangular grid, (g) hexagonal grid. **Middle column:** Reconstructions with EED-inpainting using the masks (a,d,g). **Right column:** Similar to middle column, but using optimal tonal data.



	mask	reconstruction	reconstruction with optimal tonal data
probabilistic sparsification			
MSE		24.20	14.53
+ non-local pixel exchange			
MSE		12.62	10.84

**Figure 4.15:** Evaluation of EED with different inpainting data for the test image *trui* (see Figure 4.2(a)) using 4% of all pixels. **Left column:** Different masks obtained by (a) probabilistic sparsification ( $p_{\text{cand}} = 0.05$ ,  $p_{\text{rm}} = 0.000001$ ), (d) non-local pixel exchange ( $m = 30$ , 500,000 iterations) applied to (a). **Middle column:** Reconstructions with EED-inpainting using the masks (a,d). **Right column:** Similar to middle column, but using optimal tonal data.



**Figure 4.16:** Comparison of the best results (i.e. the ones obtained by probabilistic sparsification combined with non-local pixel exchange and additional grey value optimisation) for each operator for the test image *trui* with 4% of all pixels. **Left column:** Different masks obtained with (c) homogeneous diffusion inpainting, (f) biharmonic operator, (i) EED. **Middle column:** (a) Original and (d,g,j) reconstructions using the masks (c,f,i) and optimised tonal data. **Right column:** Zoom into (a,d,g,j).

**Table 4.5:** Parameter analysis for the number of **Gauss-Seidel iterations**  $n_{GS}$  after  $n_{GN} = 20$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 1, 10, 20, and 100, we list the best result and the corresponding values for the other parameters. The column  $n_{GN}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{GN}^{\text{best}}$	$n_{GS}$	$\delta$	$\omega$	MSE
randomly selected	17	1	10.00	1.000	93.99
	<b>20</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>92.35</b>
	20	30	10.00	0.010	93.03
	20	100	10.00	0.010	92.50
rectangular grid	16	1	10.00	1.000	62.90
	18	10	10.00	0.100	62.75
	19	30	10.00	0.100	62.75
	<b>16</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>62.72</b>
hexagonal grid	20	1	10.00	1.000	61.45
	19	10	10.00	0.100	61.45
	<b>18</b>	<b>30</b>	<b>10.00</b>	<b>0.100</b>	<b>61.43</b>
	19	100	10.00	0.100	61.46
probabilistic sparsification	14	1	10.00	1.000	14.85
	<b>19</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>14.61</b>
	15	30	10.00	0.100	14.67
	18	100	10.00	0.100	14.62
non-local pixel exchange	20	1	10.00	1.000	10.88
	19	10	10.00	0.100	10.87
	<b>20</b>	<b>30</b>	<b>10.00</b>	<b>0.100</b>	<b>10.86</b>
	<b>20</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>10.86</b>

**Table 4.6:** Parameter analysis for the **derivative parameter**  $\delta$  after  $n_{\text{GN}} = 20$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 0.01, 0.1, 1, 10, and 100, we list the best result and the corresponding values for the other parameters. The column  $n_{\text{GN}}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{\text{GN}}^{\text{best}}$	$n_{\text{GS}}$	$\delta$	$\omega$	MSE
randomly selected	20	30	0.01	1.000	102.09
	19	10	0.10	0.100	98.74
	20	100	1.00	0.010	95.33
	<b>20</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>92.35</b>
	10	100	100.00	0.010	106.11
rectangular grid	19	10	0.01	0.100	64.74
	20	100	0.10	0.010	63.01
	14	10	1.00	0.100	62.78
	<b>16</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>62.72</b>
	4	1	100.00	1.000	68.27
hexagonal grid	20	10	0.01	1.000	63.77
	20	30	0.10	0.100	62.19
	20	10	1.00	0.100	61.58
	<b>18</b>	<b>30</b>	<b>10.00</b>	<b>0.100</b>	<b>61.43</b>
	5	100	100.00	1.000	66.61
probabilistic sparsification	20	30	0.01	1.000	23.85
	20	100	0.10	1.000	17.73
	19	30	1.00	1.000	15.14
	<b>19</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>14.61</b>
	20	1	100.00	0.100	15.94
non-local pixel exchange	0	100	0.01	0.001	12.62
	19	100	0.10	1.000	12.10
	19	30	1.00	1.000	10.90
	<b>20</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>10.86</b>
	5	10	100.00	0.100	11.31

**Table 4.7:** Parameter analysis for the **damping parameter**  $\omega$  after  $n_{\text{GN}} = 20$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 0.001, 0.01, 0.1, and 1, we list the best result and the corresponding values for the other parameters. The column  $n_{\text{GN}}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{\text{GN}}^{\text{best}}$	$n_{\text{GS}}$	$\delta$	$\omega$	MSE
randomly selected	20	100	10.00	0.001	100.16
	20	100	10.00	0.010	92.50
	<b>20</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>92.35</b>
	17	1	10.00	1.000	93.99
rectangular grid	20	100	1.00	0.001	69.58
	20	30	1.00	0.010	62.99
	<b>16</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>62.72</b>
	16	1	10.00	1.000	62.90
hexagonal grid	20	100	1.00	0.001	66.87
	20	100	1.00	0.010	61.79
	<b>18</b>	<b>30</b>	<b>10.00</b>	<b>0.100</b>	<b>61.43</b>
	20	1	10.00	1.000	61.45
probabilistic sparsification	19	100	10.00	0.001	18.22
	19	100	10.00	0.010	14.98
	<b>19</b>	<b>10</b>	<b>10.00</b>	<b>0.100</b>	<b>14.61</b>
	3	100	10.00	1.000	14.78
non-local pixel exchange	20	100	100.00	0.001	12.56
	19	30	10.00	0.010	11.35
	<b>20</b>	<b>100</b>	<b>10.00</b>	<b>0.100</b>	<b>10.86</b>
	9	100	10.00	1.000	10.87

**Table 4.8:** Parameter analysis for the number of **Gauss-Seidel iterations**  $n_{GS}$  after  $n_{GN} = 1000$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 1, 10, 20, and 100, we list the best result and the corresponding values for the other parameters. The column  $n_{GN}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{GN}^{\text{best}}$	$n_{GS}$	$\delta$	$\omega$	MSE
randomly selected	439	1	10.00	0.100	91.09
	688	10	10.00	0.100	91.09
	<b>816</b>	<b>30</b>	<b>10.00</b>	<b>0.010</b>	<b>90.83</b>
	<b>883</b>	<b>100</b>	<b>10.00</b>	<b>0.010</b>	<b>90.83</b>
rectangular grid	<b>840</b>	<b>1</b>	<b>1.00</b>	<b>0.100</b>	<b>62.34</b>
	669	10	1.00	0.010	62.39
	722	30	1.00	0.010	62.43
	613	100	1.00	0.010	62.39
hexagonal grid	985	1	1.00	0.100	61.23
	650	10	1.00	0.010	61.22
	<b>573</b>	<b>30</b>	<b>1.00</b>	<b>0.010</b>	<b>61.21</b>
	340	100	1.00	0.010	61.23
probabilistic sparsification	150	1	10.00	0.100	14.57
	1000	10	1.00	0.100	14.57
	493	30	1.00	0.100	14.55
	<b>320</b>	<b>100</b>	<b>1.00</b>	<b>0.100</b>	<b>14.53</b>
non-local pixel exchange	518	1	1.00	1.000	10.86
	293	10	1.00	1.000	10.86
	<b>172</b>	<b>30</b>	<b>1.00</b>	<b>1.000</b>	<b>10.84</b>
	24	100	10.00	0.100	10.85

**Table 4.9:** Parameter analysis for the **derivative parameter**  $\delta$  after  $n_{\text{GN}} = 1000$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 0.01, 0.1, 1, 10, and 100, we list the best result and the corresponding values for the other parameters. The column  $n_{\text{GN}}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{\text{GN}}^{\text{best}}$	$n_{\text{GS}}$	$\delta$	$\omega$	MSE
randomly selected	995	100	0.01	1.000	97.17
	778	30	0.10	0.100	95.26
	929	10	1.00	0.100	91.71
	<b>816</b>	<b>30</b>	<b>10.00</b>	<b>0.010</b>	<b>90.83</b>
	522	1	100.00	0.100	105.35
rectangular grid	555	10	0.01	0.100	63.01
	938	100	0.10	0.010	62.53
	<b>840</b>	<b>1</b>	<b>1.00</b>	<b>0.100</b>	<b>62.34</b>
	712	1	10.00	0.100	62.56
	4	1	100.00	1.000	68.27
hexagonal grid	880	10	0.01	0.100	62.09
	995	100	0.10	0.010	61.40
	<b>573</b>	<b>30</b>	<b>1.00</b>	<b>0.010</b>	<b>61.21</b>
	472	1	10.00	1.000	61.35
	94	100	100.00	1.000	66.57
probabilistic sparsification	991	100	0.01	1.000	18.88
	955	10	0.10	1.000	15.01
	<b>320</b>	<b>100</b>	<b>1.00</b>	<b>0.100</b>	<b>14.53</b>
	35	100	10.00	0.100	14.56
	36	1	100.00	0.100	15.83
non-local pixel exchange	995	100	0.01	1.000	12.46
	971	100	0.10	1.000	11.00
	<b>172</b>	<b>30</b>	<b>1.00</b>	<b>1.000</b>	<b>10.84</b>
	24	100	10.00	0.100	10.85
	5	10	100.00	0.100	11.31

**Table 4.10:** Parameter analysis for the **damping parameter**  $\omega$  after  $n_{\text{GN}} = 1000$  Gauss-Newton iterations using the masks from the Figures 4.14 and 4.15: For each of the tested values 0.001, 0.01, 0.1, and 1, we list the best result and the corresponding values for the other parameters. The column  $n_{\text{GN}}^{\text{best}}$  denotes the Gauss-Newton iteration at which the result was obtained. The best result regarding each mask is printed in bold face.

parameter	$n_{\text{GN}}^{\text{best}}$	$n_{\text{GS}}$	$\delta$	$\omega$	MSE
randomly selected	947	100	1.00	0.001	92.24
	<b>816</b>	<b>30</b>	<b>10.00</b>	<b>0.010</b>	<b>90.83</b>
	439	1	10.00	0.100	91.09
	817	1	10.00	1.000	92.06
rectangular grid	1000	10	0.10	0.001	64.47
	613	100	1.00	0.010	62.39
	<b>840</b>	<b>1</b>	<b>1.00</b>	<b>0.100</b>	<b>62.34</b>
	179	1	10.00	1.000	62.74
hexagonal grid	997	100	0.10	0.001	62.57
	<b>573</b>	<b>30</b>	<b>1.00</b>	<b>0.010</b>	<b>61.21</b>
	985	1	1.00	0.100	61.23
	472	1	10.00	1.000	61.35
probabilistic sparsification	997	100	10.00	0.001	15.40
	133	100	10.00	0.010	14.59
	<b>320</b>	<b>100</b>	<b>1.00</b>	<b>0.100</b>	<b>14.53</b>
	567	1	10.00	1.000	14.66
non-local pixel exchange	997	1	100.00	0.001	11.45
	159	100	10.00	0.010	11.04
	24	100	10.00	0.100	10.85
	<b>172</b>	<b>30</b>	<b>1.00</b>	<b>1.000</b>	<b>10.84</b>



## 4.5 Compression Result

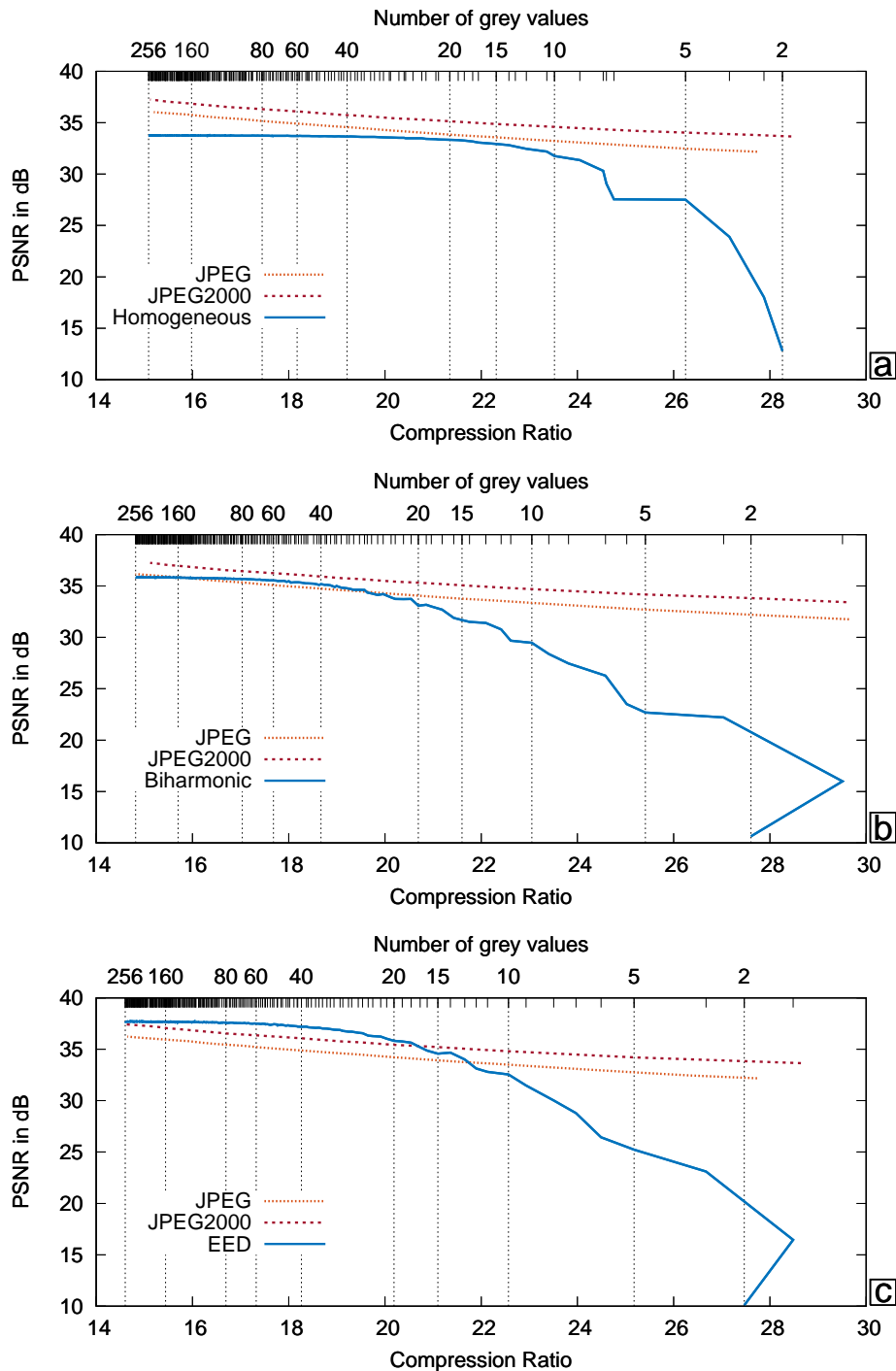
At the end of this chapter, we briefly want to get a feeling about the potential of the presented optimisation techniques from a compression point of view. To this end we simply take the qualitatively best results with respect to each interpolation operator and try to compress them with standard techniques. In Chapter 3 we have seen that *JBIG* is a suitable codec to encode binary images, whereas *LPAQ2* was our codec of choice to encode the colour values. Thus, we will rely again on those two to encode the optimised mask and the colour values, respectively. Moreover, we will not store the grey values in float precision but instead include a uniform quantisation step. This also means the stored grey values will actually not be optimal anymore, and thus the quality decreases. In reverse it allows for higher compression rates.

It also holds in general that the following results are actually not optimal from a compression point of view yet. All data have been optimised with respect to the number of pixels assuming that this number correlates with the compression ratio. However, the compression ratio also depends on the way the data is then stored. For example, let us consider a compression method for binary masks that stores the distances along the y-axis between neighbouring mask pixels whenever this distance changes. Knowing those distances, it would be possible to reconstruct the mask. For a rectangular grid we would then need to store only the distance whereas for a grid with randomly chosen locations it is likely that we would need to store the distance for every mask pixel again and again.

Thus, from a compression point of view, we would actually not be interested in the optimal data using a specific amount of mask pixels rather than the optimal data giving a specific compression rate. In this thesis we will not tackle this problem. Instead, the aim of this section is solely to elicit the general potential behind an optimal data selection for PDE-based image compression.

Figure 4.17 shows the compression results in comparison to *JPEG* and *JPEG2000* for the three operators, respectively. In addition to the compression rate that is depicted at the bottom *x*-axis, the top *x*-axis shows the corresponding number of grey values yielded by quantisation. As standard image formats anyway represent a grey value with only one byte, i.e. it can take only 256 different grey values, we choose this to be the highest number. The lowest is 2, since for only one grey value we would obtain a flat image.

As a first general observation we want to remark that the results obtained with a quantisation to 256 grey values is almost as good as the results



**Figure 4.17:** Comparison between *JPEG*, *JPEG2000* and the result based on the best spatial and tonal data (4% of all pixels) for inpainting with (a) homogeneous diffusion, (b) biharmonic interpolation and (c) EED (see also Table 4.4, bold print). The optimised spatial data has been compressed with *JBIG*, the optimal tonal data by a quantisation step followed by a compression with *LPAQ2*. The top  $x$ -axis depicts the number of grey values yielded by the quantisation whereas the bottom  $x$ -axis shows the corresponding compression ratio.

with original optimised tonal data: For homogeneous diffusion, the quality decreased from a PSNR of 33.78 to 33.76, for the biharmonic operator from 35.90 to 35.86 and for EED from 37.78 to 37.70.

Considering the whole range of quantisation levels, with homogeneous diffusion inpainting, we cannot get better results than with *JPEG*, even though we can come already close to it, when using only 20 different grey values. Using the biharmonic operator, we are able to outperform *JPEG* for compression ratios in the range 16:1 to 19:1. With EED, we are even able to compete with *JPEG2000*. For a compression ratio from 14:1 up to 20:1, the PDE-based approach is better. Best results are obtained when we use between 40 (PSNR 37.19) and 60 (PSNR 37.50) different grey values.

We also observe that the quality drastically decreases towards small numbers of different grey values. This is not really surprising, as the full range of colours of the original cannot be represented properly anymore. Thus, in order to reach higher compression rates and to maintain the quality, we should rather adapt the amount of mask pixels, i.e. using less than the 4%. The same holds if we would like to reach lower compression ratios. Here, one should increase the amount of mask pixels.

The knee at the right end of the graph in the case of EED and the biharmonic operator is related to the values being connected in the order of the quantisation values. It shows, that the reconstruction quality obtained with 2 different grey values is in this case actually better than the one with 3. The reason is that the colours are chosen by uniform quantisation. For the corresponding results, the colours obtained with two quantisation levels simply fit better than those with three.

As in Chapter 3, we could use a non-uniform quantisation to adapt the colours when the number of quantisation levels surpasses a specific value. Taking a more general perspective, this would mean we try to adapt our compression method to suit the data to be compressed better. Alternatively, one could also try to chose the data from the beginning, such that it suits the compression method better. This means we want to know what is the optimal data if only specific grey values can be chosen. As a fact, for the spatial data selection we already restrict the possible location to the one given by the pixel grid. The same could be done for the tonal data.

Carrying this further, one could restrict the locations of the mask pixels such that they can be compressed more efficiently. In a more fuzzy approach, one could favour specific locations over others with a certain probability depending on how well the location can be compressed. One of the most simple ideas would be to weight the pixel locations according to a tree structure, since such a structure can be stored very efficiently [SPM<sup>+</sup>14].

## 4.6 Conclusion

The optimality of our results in this framework refers solely to the reconstruction quality, but only indirectly includes coding costs via the amount of pixels. However, it is well known that specific pixels distributions can be encoded more efficiently than others. Examples are the tree based codecs that have been mentioned in Chapter 1 and our edge-based codec from Chapter 3.

In contrast to the previous chapter, we did not present a fully developed compression codec here. Instead, our focus was on the question of how to find the optimal interpolation data for reconstructing an image with a given PDE. To this end, we first developed two algorithms to get the optimal spatial data: The probabilistic sparsification and the non-local pixel exchange. Whereas the former creates an inpainting mask by iteratively removing insignificant pixels, the latter improves a mask that has already the desired amount of mask pixels. To obtain best results, both methods are applied consecutively. Afterwards, we compute the optimal tonal data by a least squares fit.

The results of this chapter are manifold. On one hand, we saw that even with the simplest inpainting PDE, namely homogeneous diffusion, one can obtain reconstructions of astonishing quality using only 4% of all pixels. On the other hand, we saw that the concepts and algorithms can be carried over to more sophisticated PDEs such as the biharmonic interpolation and EED. Even though the optimisation has the less influence the more sophisticated the PDE, it is still possible to improve the quality in all cases remarkably.

Furthermore, we got an impression on the potentials of optimal data selection for image compression. By applying our default compression schemes on the data obtained for EED, we are able to outperform *JPEG2000* for a specific range of compression ratios without any additional efforts. The obtained compression rates are in the rather low range, i.e. between 15:1 and 20:1. Since they have been obtained for a fixed amount of pixels, we expect that one could easily reach other ranges by choosing the amount of pixels accordingly. Since so far, PDE-based compression methods could mainly compete with *JPEG2000* on medium and high compression ratios, our results show that a method based on optimally selected data can close this gap.

Moreover, we indicated that the presented results are likely to be improved when the subsequent coding of the selected data is included into the optimisation approaches. Also a coupled optimisation approach that considers spatial and tonal data at the same time could lead to improvements. Last but not least, as we will see in the next section, the results are of interest for already existing PDE-based compression methods.

# Chapter 5

---

## Edge-based Image Compression revisited

### 5.1 Motivation

In the previous chapter, we have seen the potential of optimal data selection for PDE-based image compression. Besides storing the raw optimised data as done at the end of the previous chapter, we can also exploit the gained knowledge to improve already existing PDE-based compression methods. For example, by optimising the colour values, we can instantly increase the quality of the results of the presented edge-based codec (see Chapter 3). Of course, this will slow down the encoding time. However, in many applications like video compression, encoding time is not heavily restricted, while decoding should be possible in realtime.

The aim of this chapter is to evaluate how optimal data can be included into our edge-based image compression codec and to evaluate its impact.

### 5.2 Integrating Optimal Tonal Data

#### 5.2.1 Simplified Algorithm

To apply the grey value optimisation to the cartoon-like colour image from Chapter 3, we consider each channel separately. Note that with respect to the stored data, the underlying interpolation operator of the edge-based codec is not necessarily a linear one. This is due to the fact that it is composed of two subsequent interpolation steps. First, the data is linearly

interpolated along the edges. In a second step, the data is then spread into the 2D space by using homogeneous diffusion inpainting. Thus, we should actually use the grey value optimisation for nonlinear operators as presented for EED in Section 4.4.2. However, it turns out that the interpolation behaves in this case almost linearly. This is why it is here sufficient to perform only one Gauss-Newton iteration and solve the Gauss-Seidel equation system up to a specified precision (i.e. until the residual is small enough). The damping parameter  $\omega$  can be set to one as well as the derivative parameter  $\delta$ . The simplified algorithm, where  $\beta$  has been replaced by  $\mathbf{u}$ , reads as follows

**Algorithm: Grey value optimisation for edge-based compression**

***Input:***

Original image  $\mathbf{f}$ , inpainting mask  $\mathbf{C}$ .

***Initialisation:***

$\mathbf{u} := \mathbf{r}(\mathbf{C}, \mathbf{f})$  and  $\mathbf{g} := \mathbf{f}$ .

***Compute:***

For all  $i \in K$ :

    Compute the inpainting derivatives

$$\mathbf{d}_i := \mathbf{r}_{f_i}(\mathbf{C}, \mathbf{f}) \approx \mathbf{r}(\mathbf{C}, \mathbf{f} + \mathbf{e}_i) - \mathbf{u} .$$

Do

1. Set  $\mathbf{u}^{\text{old}} := \mathbf{u}$ .
2. For all  $i \in K$ :
  - a. Compute the correction term  $\gamma := \frac{\mathbf{d}_i^\top (\mathbf{f} - \mathbf{u})}{\mathbf{d}_i^\top \mathbf{d}_i}$ .
  - b. Update the pixel value  $g_i := g_i + \gamma$  and the reconstruction  $\mathbf{u} := \mathbf{u} + \gamma \cdot \mathbf{d}_i$ .

while  $|\text{MSE}(\mathbf{u}, \mathbf{f}) - \text{MSE}(\mathbf{u}^{\text{old}}, \mathbf{f})| > \varepsilon$ .

***Output:***

Optimised pixel values  $\mathbf{g}$ .

Note that this algorithm is almost identical to the one presented for the linear case in Section 4.3.2). The only difference is that we are using the inpainting derivative instead of the inpainting echo.

## 5.2.2 Results

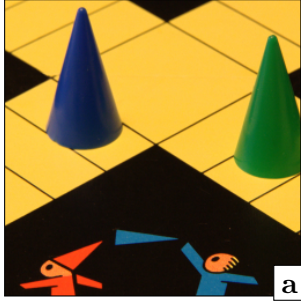

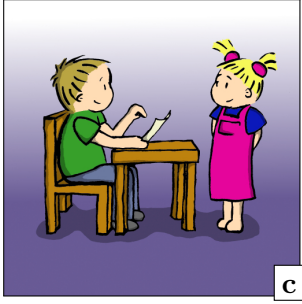
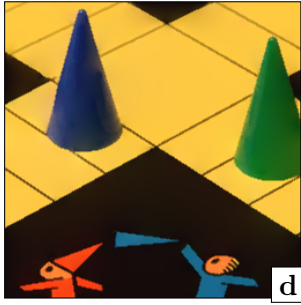


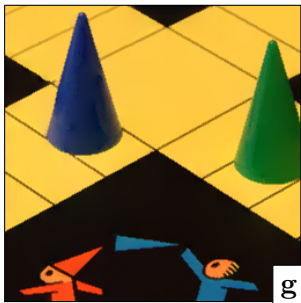


To evaluate the effects of optimised tonal data for the edge-based codec, we apply it to the same test images as in Chapter 3, Figure 3.5. All parameters have been unaltered. The results are depicted in Figure 5.1 and Table 5.1. Whilst the reconstruction quality improved for all images, we also observe that the tonal optimisation has a larger impact on the non-synthetic images *coppit* and *svalbard*. For them, the PSNR changed from 30.31 dB to 33.26 dB and 30.14 dB to 32.02 dB, respectively. For the test image *comic* it only increased from 30.20 dB to 30.91 dB.

This observation is emphasised by the fact that the bit rate for *coppit* and *svalbard* increases only slightly for the optimised colour signal, i.e. by 0.01 bpp, whereas in the case of *comic* it raises from 0.19 to 0.23 bpp.

As a consequence, the gap to *JPEG* and *JPEG2000* (see Table 5.1) to our method increases for *svalbard* and *coppit*. This is not so much the case for *comic*. Still, in all cases our method is superior to *JPEG* and *JPEG2000*.

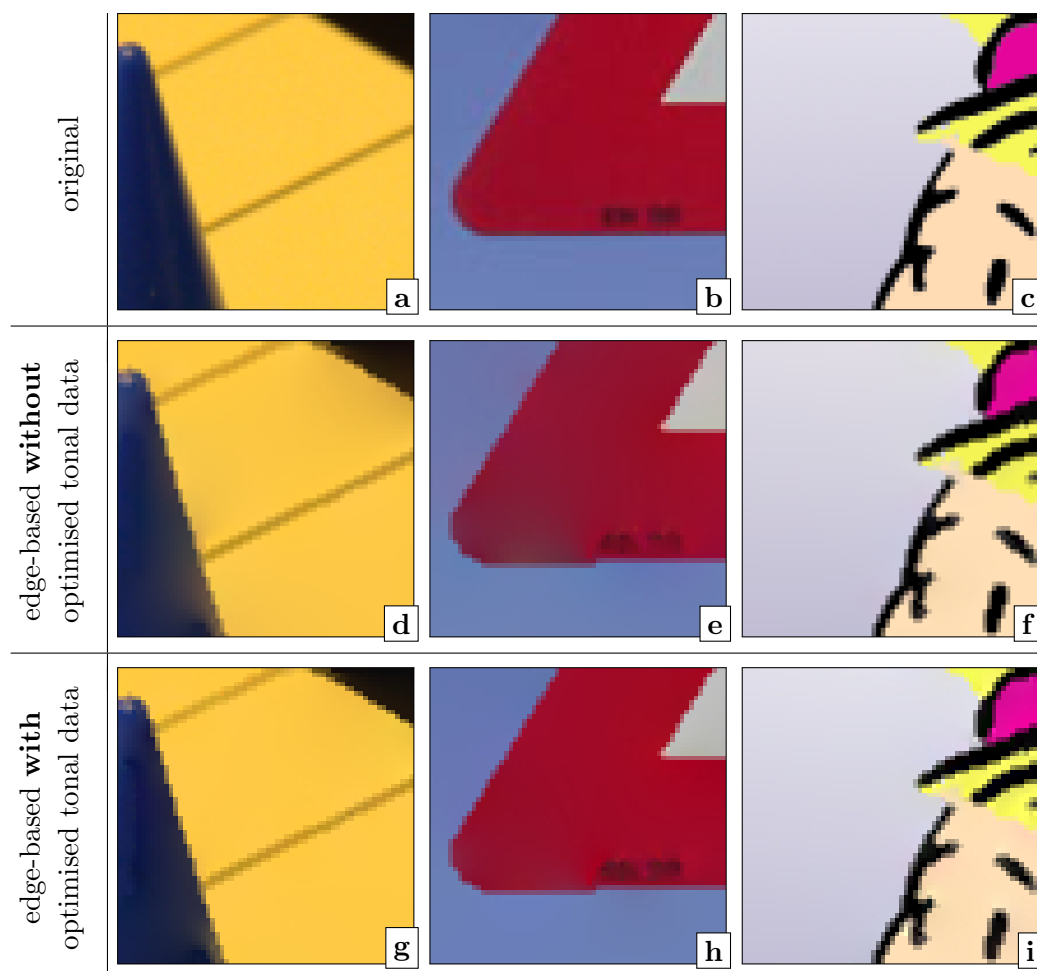
The main reason for the big improvement in the case of *svalbard* and *coppit* can be explained with Figure 5.2. It depicts a cropped detail for each result. Here one sees that choosing the colour next to the edges can lead to colour bleeding artefacts in the reconstruction (see Figure 5.2(e) and (h)). This happens because the colours next to the edges do not reflect the main colour of the whole region next to it. This can happen due to a delocalised edge, slightly unsharp edges, or artefacts in the original image. By optimising the colour values, we can overcome this problem and improve the visual reconstruction quality notably.

In Section 3.4.1 and Figure 3.7 we have seen an example with extreme compression rates. By removing low contrast edges and adapting quantisation, sampling and edge tracing parameter, we have reached a compression ratio of 315:1. Of course the quality of the reconstruction decreased to a PSNR of 28.18. Considering the visual result (see Figure 5.3(c) and (f)), it is obvious that the main reason for this reduced quality is the colour bleeding. Therefore, it is not surprising that by using optimised tonal values the quantitative and visual quality improves significantly. Quantitatively, the PSNR of 30.36 dB is even slightly better than the result from Chapter 3, Figure 3.5. There, the PSNR has been 30.14, at a rate of 0.16 bpp (ratio of approximately 145:1) instead of 0.07 bpp, (ratio of 320:1) here.

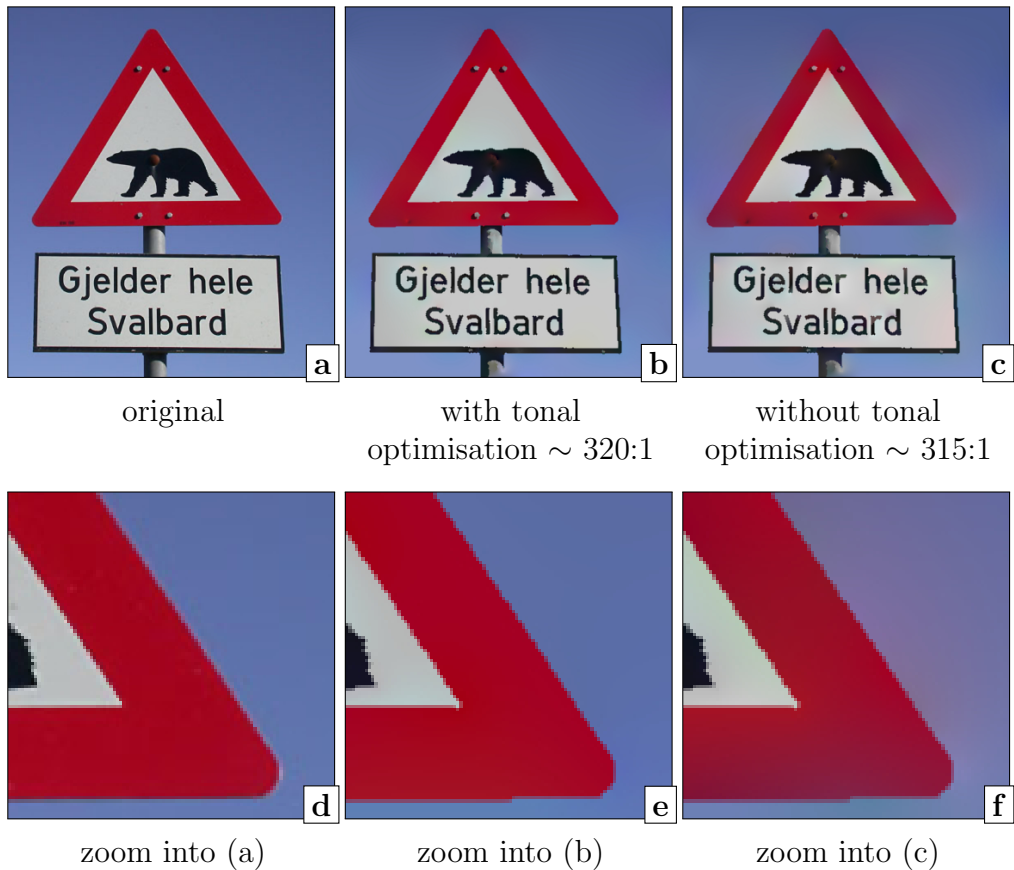
original			
edge-based <b>without</b> optimised tonal data			
PSNR	30.31 dB	30.14 dB	30.20 dB
edge-based <b>with</b> optimised tonal data			
PSNR	33.26 dB	32.02 dB	30.91 dB

**Figure 5.1:** Comparison of the edge-based codec with and without tonal optimisation for different test images. **First row:** Original images. **Second row:** Resembles the results of the last row of Figure 3.5, i.e. using our codec with default parameters and edge images as depicted in Figure 3.4; Bitrates from left to right: 0.37 bpp, 0.16 bpp and 0.19 bpp. **Third row:** Has been obtained with the same parameter settings except that the stored tonal data was optimised additionally; Bitrates from left to right: 0.38 bpp, 0.17 bpp and 0.23 bpp.





**Figure 5.2:** Cropped detail ( $64 \times 64$ ) for each image depicted in Figure 5.1.



**Figure 5.3:** Comparison of the edge-based codec with and without tonal optimisation for extreme compression ratios: **(a)** Original image; **(b, c)** Our codec using the edge image as depicted in Figure 3.7 (c) and the same settings as in Figure 3.7 (b), i.e. default settings except for  $q_* = 15$ ,  $d_* = 45$  ( $* \in \{R, G, B\}$ ) and  $d_{tr} = 10$ . For (b) the colour values have been optimised whereas for (c) not i.e. it shows Figure 3.7 (b). Both results have a compression rate of 0.07–0.08 bpp (i.e. approx. 315:1–320:1). While (b) has a PSNR of 30.36 dB, the one of (c) is 28.18 dB; **(d-f)** Zoom into (a-c).

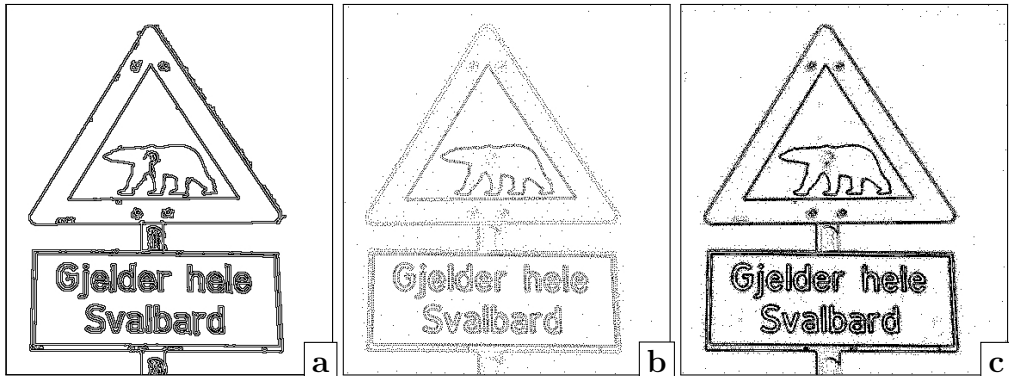
**Table 5.1:** Comparison of quantitative results (error measure PSNR in dB) to demonstrate the capabilities of tonal optimisation in the context of edge-based image compression: The first part of this table resembles the results stated in Table 3.3, i.e. the results of our codec with default parameters and edge images as depicted in Figure 3.4 in comparison to *JPEG* and *JPEG2000* at the same compression rates. The second part of the table shows the analogous results when using the same parameter settings for our codec except that the stored tonal data was optimised additionally.

image	<i>coppit</i>	<i>svalbard</i>	<i>comic</i>
<b>without</b> optimised tonal data:			
compression rate (bpp)	0.37	0.16	0.19
our method	<b>30.31</b>	<b>30.14</b>	<b>30.20</b>
<i>JPEG</i>	26.61	23.38	24.25
<i>JPEG2000</i>	28.13	27.68	26.77
<b>with</b> optimised tonal data:			
compression rate (bpp)	0.38	0.17	0.23
our method	<b>33.26</b>	<b>32.02</b>	<b>30.91</b>
<i>JPEG</i>	26.87	23.38	25.87
<i>JPEG2000</i>	28.29	27.90	28.03

### 5.3 Integrating Optimised Spatial Data

Let us now have a look on how the edge-based compression codec could benefit from the spatial optimisation approach. In contrast to tonal optimisation, it cannot be applied instantly as the spatial data is actually already prescribed by the edges. Anyway the question is why not to replace the edges and select the data solely by our optimisation approach.

For most cartoon-like images the resulting inpainting mask would actually not differ much from the mask derived by the edge image. Figure 5.4 shows two masks with different densities for the test image *svalbard* obtained with the probabilistic sparsification and the non-local pixel exchange. For comparison, the mask based on the edge image which was used to compress *svalbard* in Chapter 3 is depicted as well. For all masks, the data is mainly chosen next to the edges. However, the spatially optimised mask



**Figure 5.4:** (a) Inpainting mask derived from edge image as given in Figure 3.4(b). It can be encoded with *JBIG* with 1416 byte. (b,c) Applying the probabilistic sparsification ( $p_{\text{cand}} = 0.02$ ,  $p_{\text{rm}} = 0.000001$ ) and the non-local pixel exchange (400,000 iterations,  $m = 50$ ) to the cartoon-like image *svalbard* with a density of (a) approx. 4% and (b) approx. 10%. Those images can be encoded with *JBIG* with 3373 and 4456 bytes, respectively. Parameters for (b) and (c) have been determined as in Chapter 4.

cannot be compressed as efficiently as the pure edge image. This is because there is always a few scattered pixels in homogeneous regions and the data at edges are often not connected and more fuzzy than in our edge-based approach. Moreover, the spatial data is stored as it was obtained, whereas in the edge-based approach not the locations next to the edges but only the edges themselves need to be encoded. Last but not least, apart from a less efficient mask encoding, one cannot exploit the advantage of data reduction by subsampling the colours along the edges.

Using the mask from Figure 5.4(b) with optimal tonal data and requantising to 17 different values per channel, we achieve the same quantitative quality (PSNR 32.26) as with the edge-based codec, but need 0.43 bpp instead of only 0.38 bpp. This means we would not be able to compete with *JPEG2000*.

However, there are cases in which a *hybrid method*, i.e. a mixture of edges supplemented by a few optimally chosen points can be advantageous. Such cases are given by images that contain low contrast or blurry edges as they are given within the cones of the test image *coppit* or in natural images such as the test image *trui*. Those edges can be expected to be represented sufficiently well with a few additional optimally chosen pixels. A description with edges is in this case usually very costly or does not give satisfactory results.

### 5.3.1 Hybrid Method

As in the edge-based codec, we start by detecting the edges. However, this time we are only interested in the high contrast ones. To this end, it is usually sufficient to use the same edge parameters as before but increase  $T_1$  such that it is close to  $T_2$ . In a second step, we add a specified amount of spatially optimised pixels. This is done by first applying the probabilistic sparsification (cf. Section 4.2.2) with only a small difference: We exclude the pixels adjacent to the edges from the set  $K$ , i.e. they cannot be removed by the sparsification process. Afterwards, we apply the non-local pixel exchange (cf. Section 4.2.3), again excluding the pixels adjacent to the edges. Finally, we obtain a mask which contains the pixels adjacent to the edges and the specified amount of spatially optimised pixels.

In order to encode the mask, we could simply store it using *JBIG*. However, since it explicitly contains the pixels next to edges its demand for storage increases compared to the encoding of edges as done in the original codec. Alternatively, we could store the edge image and the additional points separately. However, ideally we would like to store edges and points in the same image with *JBIG*. This demands that during decoding we can distinguish between the points and edges.

To achieve this, we exploit that edges most often consist of more than just a single pixel, whereas the additionally added pixels are likely to not have neighbours. To guarantee this condition, we modify our codec slightly as follows. After the edge detection has been completed, we apply a post-processing step, which removes edges of length one, i.e. pixels that do not have a neighbour. For *probabilistic sparsification*, not only do we remove the pixels adjacent to edges, but also the edge pixels themselves from the initial set  $K$ . The same edge pixels are also discarded from the initial mask  $\mathbf{C}$ . Moreover, the initial set  $K$  and mask  $\mathbf{C}$  only contain pixels that lie on a grid with grid distance 2. For the *non-local pixel exchange*, we simply demand that during an exchange a mask pixel cannot be set next to another mask pixel.

Of course, by introducing those restrictions, it might happen that we prevent the pixels from being set to their true optimal location. However, experiments have shown that the gain in compression justifies this approach.

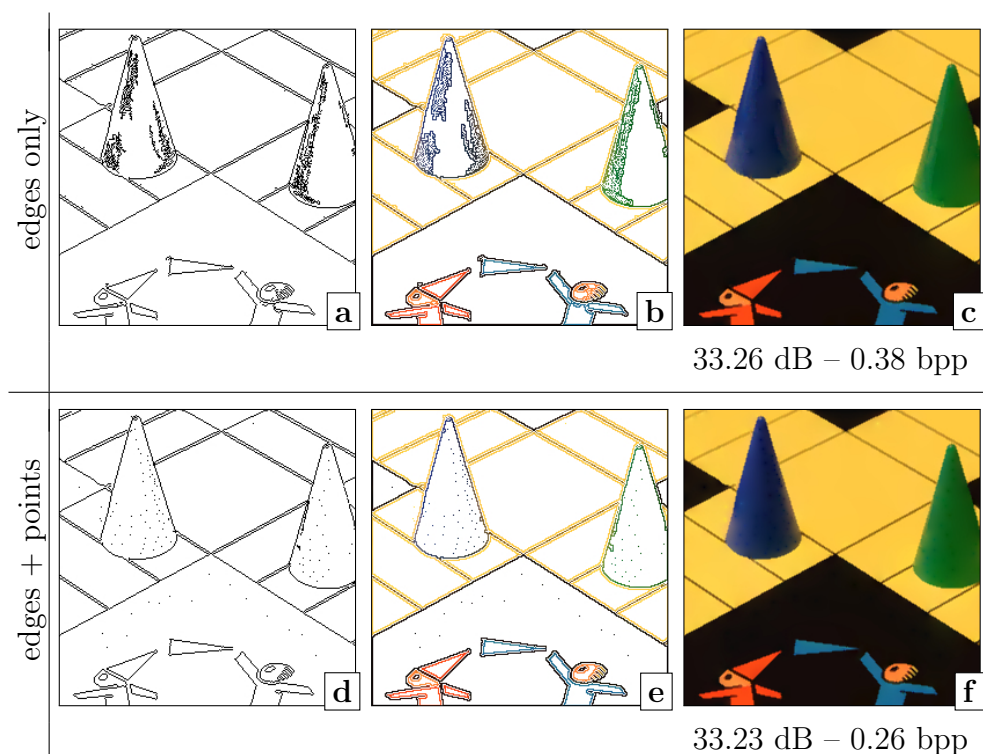
### 5.3.2 Results

As a first experiment we apply the new approach to the test image *cop-pit*. Except for the edge detection parameter  $T_1$ , we keep all parameters fixed as they were chosen for the results that are purely based on edges

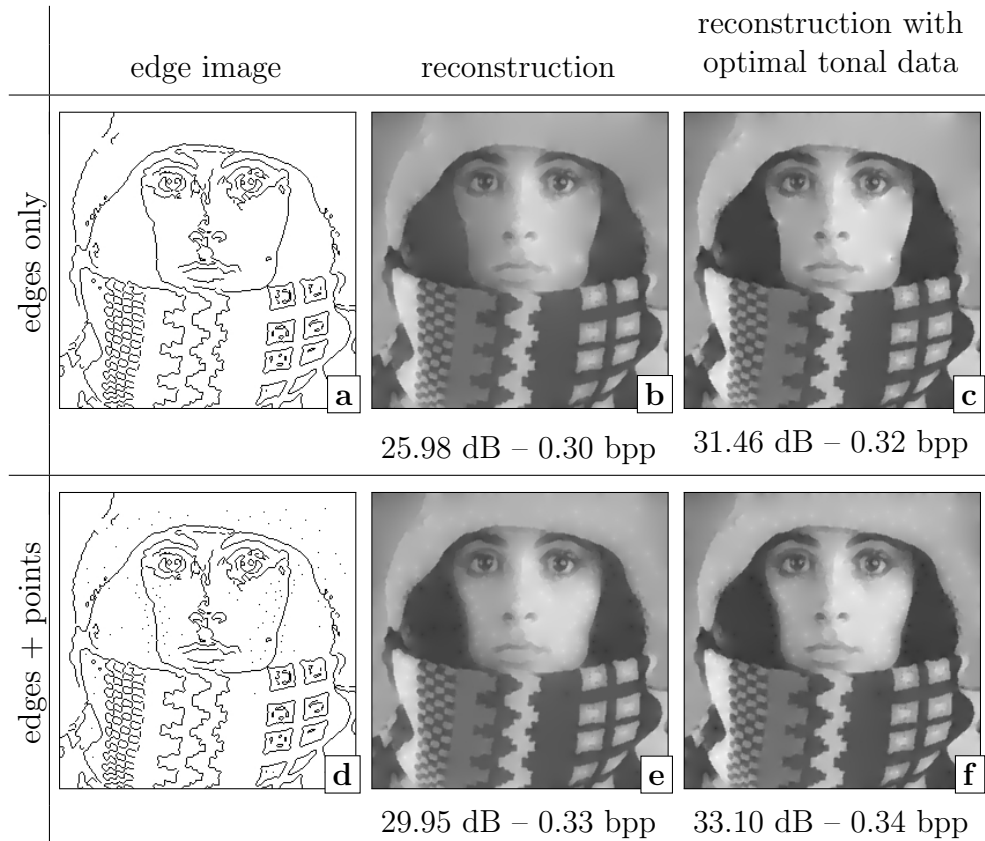
(see Chapter 3, Figure 3.5). In addition, we optimise the tonal data. By increasing the threshold  $T_1$  for the edge detection from 4.3 to 20 we reduce the amount of edges such that only the high contrast ones are left. Then, we add 100 additional pixels as described in previous section (Section 5.3.1). The resulting edge image, which is encoded by using *JBIG*, is depicted in Figure 5.5(d). As expected, the majority of the points have been put at the low contrast edges in the cones. Figure 5.5(e) shows the decoded colours placed at their positions for inpainting. The final reconstruction is given by Figure 5.5(f). For comparison, the top row of Figure 5.5 shows the results which were obtained previously with the codec purely based on edges including tonal optimisation (as Figure 5.1). In both cases, we obtain almost the same reconstruction quality, i.e. a PSNR around 33.25 dB. However, the hybrid method needs only 0.26 bpp instead of 0.38. This corresponds to a compression ratio of approximately 90:1. At the same compression rate, *JPEG2000* would have a PSNR of 25.86 dB.

As a second experiment we want to see if this approach is more suitable to encode natural images than the pure edges based one (cf. Section 3.4.2 and Figure 3.8). To this end, we first want to illustrate the effect of optimised data on the reconstruction quality for the test image *trui*. Figure 5.6 shows an edge image for the test image *trui* containing only the high contrast edges. Figure 5.6 states the corresponding reconstruction using the edge-based codec as it was defined in Chapter 3. By applying the tonal optimisation, we can improve the quantitative result greatly from a PSNR of 25.98 dB to a PSNR of 31.46 dB (see Figure 5.6(c)). However, the visual result is not really convincing, as we obtain sharp edges which look cartoon-like, and thus unnatural. On the other hand, Figure 5.6(e) depicts the result if we do not use optimal tonal data, but only include spatially optimised data, by adding 100 pixels using the hybrid method. In this case the quality can also be increased considerably from 25.98 dB to 29.95 dB. The best result is presented in Figure 5.6(f) where tonally and spatially optimised data is included and we end up with a PSNR of 33.10 dB.

Finally, we want to see if the hybrid method is competitive to *JPEG* and *JPEG2000*. Moreover, we are interested if it is better for this type of images than the approach based solely on edges or solely on points. To this end, Figure 5.7 compares the quality of all methods for the test image *trui* at a compression rate of 0.34 bpp. All parameters have been tuned to deliver optimal results. Quantitatively, *JPEG2000* delivers the best result. Our hybrid method is slightly better than *JPEG*. The point based approach (cf. Section 4.5) is worse, but still better than the edge-based one (cf. Chapter 3). Visually, *JPEG*, *JPEG2000*, and the hybrid method are quite comparable. The approach which is purely based on edges looks unnatural, cartoon-like, while the one purely based on points looks blurry.

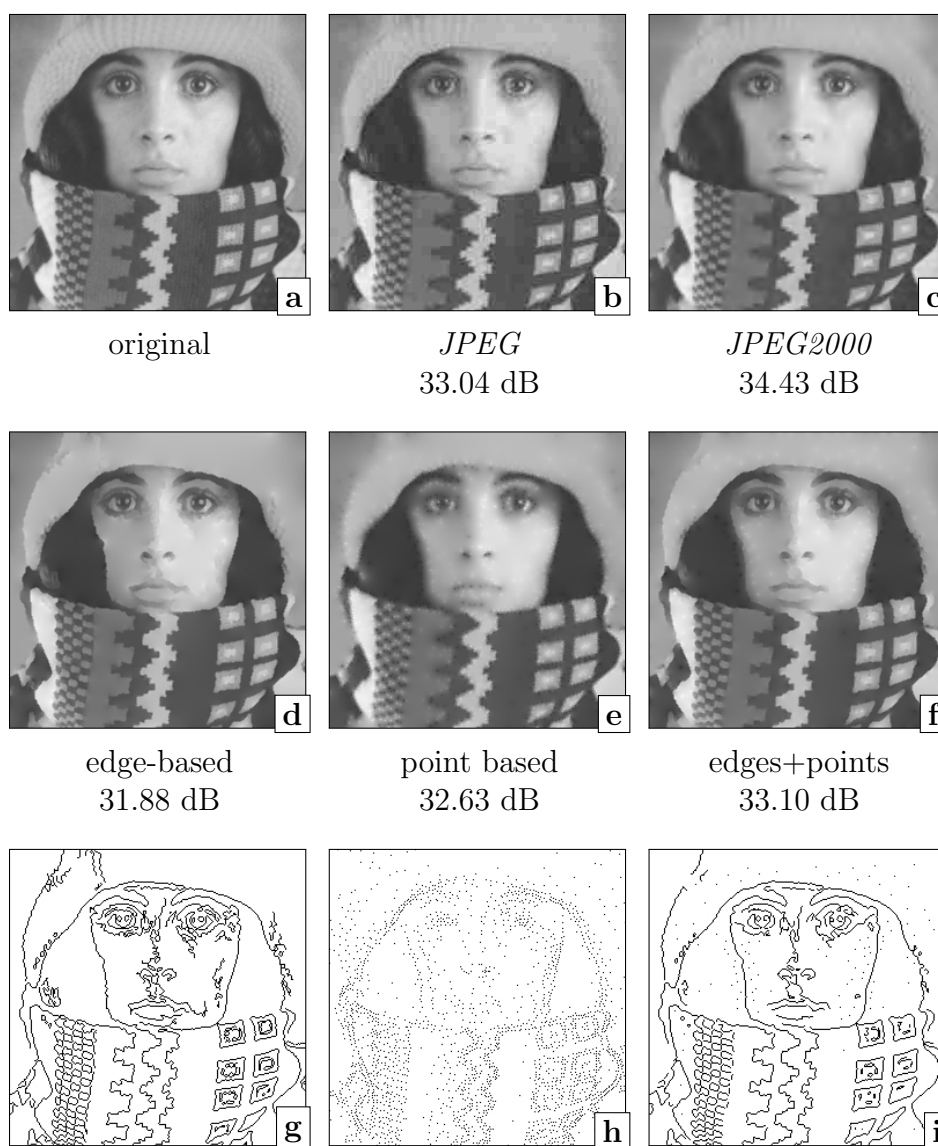


**Figure 5.5:** Comparison of the original edge-based codec with the version which allows to add spatially optimised points. In both cases the colour values have been optimised. **From left to right:** Underlying edge images, reconstructed colours, decoded images. **Top row:** Results from Figure 5.1, i.e. with edges only, having a PSNR of 33.26 dB and 0.38 bpp. **Bottom row:** Result with a reduced amount of edges by setting  $T_1 = 20$ , but including 100 spatially optimised points. Has roughly the same quality (PSNR of 33.23 dB), but needs only 0.26 bpp. Parameters for spatially optimised points: probabilistic sparsification:  $p_{\text{cand}} = 0.1$ ,  $p_{\text{rm}} = 0.1$ , non-local pixel exchange: 1,000 iterations,  $m = 20$ . Other parameters have been chosen identically in both approaches.



**Figure 5.6:** Effect of adding spatially optimised points to a given edge image. (a) Edge image obtained by zero-crossings-based edge detection ( $T_1 = 3.5$ ,  $T_2 = 5$ ,  $\sigma = 1.5$ ). (b) Result of the edge-based codec using the edge image (a) and default settings except for  $q_* = 24$ ,  $d_* = 5$ . (c) As (b) but with optimal tonal data. (d) Adding 100 points to (a) with probabilistic sparsification ( $p_{\text{cand}} = 0.05$ ,  $p_{\text{rm}} = 0.000001$ ) and non-local pixels exchange (100,000 iterations,  $m = 20$ ). (e) Decoded image based on (d) and the same parameters as in (b). (f) As (e) but with optimal tonal data.





**Figure 5.7:** Comparison of compression methods at 0.34 bpp. For the PDE-based methods colour values have been optimised. (a) Original image. (b) *JPEG*. (c) *JPEG2000*. (d) Result of the edge-based codec using the edge image (g) and default settings except for  $q_* = 20$ . (e) Inpainting result using the mask (h) and quantising to  $q_* = 21$  different grey values (cf. Section 4.5). (f) Similar to Figure 5.6(f). (g) Edge image ( $T_1 = 2$ ,  $T_2 = 5$ ,  $\sigma = 1.2$ ). (h) Mask with 2300 pixels obtained by probabilistic sparsification ( $p_{\text{cand}} = 0.05$ ,  $p_{\text{rm}} = 0.000001$ ) and non-local pixels exchange (500,000 iterations,  $m = 20$ ). (i) As in Figure 5.6(d).

## 5.4 Conclusion

In this chapter, we have seen how optimal data can be integrated into our edge based codec from Chapter 3. In the case of the tonal optimisation, we could use a simplified version of the grey value optimisation for the non-linear case, as introduced for EED in Chapter 4. The improved codec allowed to improve the results drastically. This clearly indicates that any kind of PDE-based compression codec should include optimised tonal data as long as encoding time is not a limiting factor.

In the case of spatial data we supplemented the edge images by only a few spatially optimised pixels. Therefore, we adapted the algorithms of Chapter 4 and embedded them into the edge-based codec. As a result the codec is able to represent smooth variations and blurry edges much more appropriately. In cartoon-like images improvements can be achieved by tuning the edge detection parameters such that low contrast edges are no longer detected but replaced by additional optimised spatial data. Also non-cartoon-like images can now be encoded in good quality. The result for the test image *trui* is similar to the one obtained with *JPEG* and better than the ones that are based on edges or optimal data only. Although we could not outperform *JPEG2000*, this result shows, that even with homogeneous diffusion inpainting results of good quality can be obtained by combining different concepts of PDE-based image compression.

# Chapter 6

---

## Summary and Outlook

### 6.1 Summary

In this thesis we investigated PDE-based image compression based on edges and optimal data. Starting with an introduction and motivation we first established the basis of PDE-based compression by explaining PDE-based inpainting in Chapter 2. We have analysed the numerical theory behind inpainting with homogeneous diffusion and have provided a fast multigrid algorithm for solving the inpainting problem. Moreover, we discussed the extension to more sophisticated differential operators.

Based upon this knowledge, we developed in Chapter 3 a PDE-based compression codec that is able to outperform *JPEG* and *JPEG2000* for cartoon-like images. By extracting edges and the adjacent pixel values, by encoding them efficiently, and by using homogeneous diffusion for reconstruction, we have created a conceptually simple, yet not less efficient compression method.

In the case of our edge-based codec, the decision about which data should be stored was based upon perceptual relevant image features. However, the striving for perfection and the highest efficiency raises the more fundamental question which data is actually optimal in terms of reconstruction quality. Therefore, we have explored in Chapter 4 how to find optimal data for given PDEs. We have shown that even for the simplest inpainting PDE, namely homogeneous diffusion, one can obtain reconstructions of astonishing quality using only 4% of all pixels. However, this requires to optimise the data carefully in both the domain and the co-domain. For this purpose, we have suggested two algorithms, the probabilistic sparsifica-

tion and the non-local pixel exchange. Applying them consecutively allows to outperform the quality of the approach of Belhachmi et al. [BBBW09]. Thus, it proves that Belhachmi’s optimality result only applies to the continuous setting, and that it cannot easily be transferred to the discrete setting without admitting additional errors. Regarding the tonal optimisation, we have proposed a least squares approach that allows to compute the exact optimal grey values for a given inpainting mask.

To evaluate the potential of an optimal data selection for PDE-based image compression, we have encoded the obtained data by straightforward methods as introduced for the edge-based codec. Even with this simple approach, the result obtained with EED can already outperform the quality of *JPEG2000*.

In Chapter 5 we have shown that an optimal data selection can also be of interest for already existing compression codecs. By optimising the tonal data in our edge-based codec, we could instantly achieve remarkable improvements regarding the quality of the results. Furthermore, we have shown that for images that contain blurry edges, it can be beneficial to add spatially optimised data next to the edges. This hybrid method allowed us to achieve also for non-cartoon-like images good qualities using the simplest PDE, homogeneous diffusion. In fact, the results are better than the results obtained with single pixels or edges only and are close to the quality of *JPEG2000*.

## 6.2 Conclusion and Outlook

The results of this thesis are manifold. On one hand, we presented a competitive codec that is tailored for the compression of cartoon-like images. On the other hand, we introduced approaches how to select the optimal data for reconstruction in the general case. Both topics teach us that homogeneous diffusion inpainting should not categorically be dismissed for PDE-based image compression, albeit it has a bad reputation for inpainting tasks. We saw that depending on the selected data, it might lead to high quality results. At the same time, we can benefit from its simplicity and speed.

However, this fundamental insight does not contradict the favourable position of EED for PDE-based image compression. After all, it has delivered the best quality for a fixed amount of data in Chapter 4. Also, we could observe that EED is least sensitive to a good spatial data selection and is thus well-suited when the data selection is restricted to specific positions.

Furthermore, the results of our edge-based codec indicate that it is often necessary to adapt to specific types of images. Cartoon-like images are one type that clearly needs a specialised treatment and PDE-based methods seem to be well suited to fulfil this task.

We also saw that there is a high potential of an optimal data selection for PDE-based image compression. Using straightforward methods to encode the obtained data, we could outperform *JPEG2000* at low compression rates. Since so far, PDE-based compression methods could mainly compete with *JPEG2000* on medium and high compression ratios, our results show that a method based on optimally selected data can close this gap.

Moreover, even if optimal data alone might not always lead to competitive results, it can improve existing codecs and help to overcome specific problems. Our edge-based codec was improved substantially by simply optimising the tonal data. By including some spatial points, smooth edges could be represented.

Finally, we consider our contributions to open the door for various further interesting research topics. Regarding our edge-based image compression codec, the development of progressive modes could be one of those. The main challenge would possibly lie in the progressive transmission of the edge image. Since we are encoding the edge image with *JBIG*, its progressive mode might offer a point of entry.

As already mentioned, there have been extensions based on our edge-based codec in order to suit the compression of depth maps [LSJO12, GMG12]. In [HMWP13], even optimal data was included, similar to our hybrid method. Thus, another interesting area of research would be to analyse its adaptation to further image types that also share specific characteristics with cartoon-like images. One of those types might be medical images.

Furthermore, real-time video compression for animated cartoon movies as well as extensions to 3D image data seems to be a feasible research topic (cf. [Lun11]).

Since diffusion curves [OBW<sup>+</sup>08] have not been optimised and used for image compression, yet, one could also investigate how this can be achieved. Besides the representation of edges with splines, an interesting aspect which also appears in [Eld99] is the representation of smooth edges. In contrast to our hybrid method, where additional spatial and tonal data is added to represent the smoothness effect, his method creates it by storing additional information explicitly at the edge. It is likely that this information can be encoded more efficiently than extra pixels.

As we have seen in our hybrid method in Chapter 5, it is not hard to include optimal data into existing methods (see also [HMWP13]). Especially the algorithms for spatial optimisation easily carry over to any kind

of interpolant. Thus, we can imagine that various approaches might benefit if a few additional data is added to improve the quality. In this context it would also be interesting to combine the capabilities and advantages of different differential operators. For instance, one could use homogeneous diffusion inpainting as suggested for the edge codec. To improve the quality then further, the difference between original and current reconstruction could be computed. This difference image again could be encoded using optimal data and EED.

Regarding PDE-based image compression with optimal data, there are several concerns that deserve attention for future work. As seen in Chapter 4, we were already able to outperform the quality of *JPEG2000* using EED and common encoding techniques. Thus, one could address the problem how these data can be encoded more efficiently. This might allow to obtain even with homogeneous diffusion results that are competitive with *JPEG2000*. On the other side one could try to improve the optimisation process with respect to compression. That means not only the reconstruction quality should determine the optimality of a pixel position or value but also the resulting coding costs. Regarding the spatial component, an easy approach to tackle this problem would be to weight locations according to their coding cost. The weight of each pixel is then converted to a probability which can be used for a biased candidate selection in the probabilistic sparsification and non-local pixel exchange. Furthermore, appropriate adaptations of the grey value optimisation to quantised data should be included.

Also, a coupled optimisation approach that considers spatial and tonal data at the same time could lead to improvements.

Another issue is the fairly long running time of our algorithms. For practical purposes one would not run our algorithm with the optimal parameters, but instead adapt them to obtain solutions of sufficient quality more quickly. Still, they would probably take minutes or even hours. As a remedy, methods as suggested in [HSW13] could increase the overall speed of the method.

One of the most promising fields connected to our probabilistic sparsification process are progressive image compression modes. As probabilistic sparsification naturally creates masks of different densities that are according to the algorithm all optimal, it already allows some kind of scalability. Even more suited is the probabilistic densification of [HMWP13] which is based on the idea of probabilistic sparsification. Instead of sparsifying the mask it fills it up instead. Thus, this method would even allow to transmit a progressive version of the image while the encoding process is still going on.

Last but not least, we believe that ongoing research in PDE-based image compression should not only prescribe specific grey or colour values at specific locations but include additional information that influence the PDE. For instance, it is conceivable to prescribe a fixed diffusion tensor at specific locations, which may encode diffusivities and directional information.

Considering the related work on PDE-based image compression stated in Chapter 1, and the open research topics, it becomes clear that our contributions represent only small pieces in the big picture of PDE-based image compression research. Each work spotlights on different facets, allowing superior results in its domain. This does not only show the flexibility and wide range of fields of application for PDE-based image compression but also gives hope that by combining all aspects, one day the outcome will be a comprehensive PDE-based image compression codec.





---

# Bibliography

## Own Publications

---

- [HMWP13] S. Hoffmann, M. Mainberger, J. Weickert, and M. Puhl. Compression of depth maps with segment-based homogeneous diffusion. In A. Kuijper, T. Pock, K. Bredies, and H. Bischof, editors, *Scale-Space and Variational Methods in Computer Vision*, volume 7893 of *Lecture Notes in Computer Science*, pages 319–330. Springer, Berlin, June 2013.
- [Mai08] M. Mainberger. Contour coding for PDE-based image reconstruction. Master’s Thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany, 2008.
- [MBWF11] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. Edge-based image compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition*, 44(9):1859–1873, September 2011.
- [MHW<sup>+</sup>12] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann, and B. Doerr. Optimising spatial and tonal data for homogeneous diffusion inpainting. In A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 26–37. Springer, Berlin, June 2012.

- [MSB<sup>+</sup>12] M. Mainberger, C. Schmaltz, M. Berg, J. Weickert, and M. Backes. Diffusion-based image compression in steganography. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, C. Fowlkes, S. Wang, M.-H. Choi, S. Mantler, J. Schulze, D. Acevedo, K. Mueller, and M. Papka, editors, *Advances in Visual Computing (Part II)*, volume 7432 of *Lecture Notes in Computer Science*, pages 219–228. Springer, Berlin, July 2012.
- [MW09] M. Mainberger and J. Weickert. Edge-based image compression with homogeneous diffusion. In X. Jiang and N. Petkov, editors, *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 476–483. Springer, Berlin, 2009.
- [SMMW13] C. Schmaltz, N. Mach, M. Mainberger, and J. Weickert. Progressive modes in pde-based image compression. In *Picture Coding Symposium*, pages 8–11, San Jose, CA, May 2013. IEEE Computer Society Press.
- [SPM<sup>+</sup>13] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert, and A. Bruhn. Understanding, optimising, and extending data compression with anisotropic diffusion. Technical Report 329, Department of Mathematics, Saarland University, Saarbrücken, Germany, March 2013.
- [SPM<sup>+</sup>14] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert, and A. Bruhn. Understanding, optimising, and extending data compression with anisotropic diffusion. *International Journal of Computer Vision*, 108(3):222–240, July 2014.
- [VBMW12] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. Dense versus sparse approaches for estimating the fundamental matrix. *International Journal of Computer Vision*, 96(2):212–234, January 2012.

## Other References

---

- [AD96] V. Aurich and U. Daub. Bilddatenkompression mit geplanten Verlusten und hoher Rate. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Proc. 18. DAGM-Symposium*, Informatik Aktuell, pages 138–146, Heidelberg, Germany, September 1996. Springer, Berlin.

- [ADF05] F. Alter, S. Durand, and J. Froment. Adapted total variation for artifact free decompression of JPEG images. *Journal of Mathematical Imaging and Vision*, 23(2):199–211, September 2005.
- [AG94] T. Acar and M. Gökmen. Image coding using weak membrane model of images. In A. K. Katsaggelos, editor, *Visual Communications and Image Processing '94*, volume 2308 of *Proceedings of SPIE*, pages 1221–1230. SPIE Press, Bellingham, 1994.
- [Aro67] G. Aronsson. Extension of functions satisfying Lipschitz conditions. *Arkiv för Matematik*, 6(6):551–561, June 1967.
- [BBBW09] Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert. How to choose interpolation data in images. *SIAM Journal on Applied Mathematics*, 70(1):333–352, 2009.
- [BGS05] A. Borzi, H. Grossauer, and O. Scherzer. Analysis of iterative methods for solving a Ginzburg-Landau equation. *International Journal of Computer Vision*, 64(2–3):203–219, 2005.
- [BHH<sup>+</sup>98] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. Le Cun. High quality document image compression with DjVu. *Journal of Electronic Imaging*, 7(3):410–425, July 1998.
- [BHM00] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, second edition, 2000.
- [BHN98] A. M. Bruckstein, R. J. Holt, and A. N. Netravali. Holographic representations of images. *IEEE Transactions on Image Processing*, 7(11):1583–1597, November 1998.
- [Bjö96] A. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.
- [BL77] Jr. B.F. Logan. Information in the zero crossings of band-pass signals. *Bell Systems Technical Journal*, 56:487–510, April 1977.

- [BM07] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, July 2007.
- [BPC09] S. Bougleux, G. Peyré, and L. Cohen. Image compression with anisotropic triangulations. In *Proc. Tenth International Conference on Computer Vision*, Kyoto, Japan, October 2009.
- [Bra77] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, April 1977.
- [Bru10a] A. Bruhn. Efficient numerical solvers for pde-based image inpainting. Personal communication, 2010.
- [BSCB00a] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. SIGGRAPH 2000*, pages 417–424, New Orleans, LA, July 2000.
- [Buh03] M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, Cambridge, UK, 2003.
- [BW10] E. Bae and J. Weickert. Partial differential equations for interpolation and compression of surfaces. In M. Daehlen, M. Floater, T. Lyche, J.-L. Merrien, K. Mørken, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 1–14. Springer, Berlin, 2010.
- [BWF<sup>+</sup>03] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In N. Petkov and M. A. Westenberg, editors, *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 222–229. Springer, Berlin, 2003.
- [Cai88] L. D. Cai. Some notes on repeated averaging smoothing. In J. Kittler, editor, *Pattern Recognition*, volume 301 of *Lecture Notes in Computer Science*, pages 597–605. Springer, Berlin, 1988.
- [Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

- [CBAB97] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311, 1997.
- [Che87] S. Chen. Image reconstruction from zero-crossings. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, volume 2, pages 742–744, Milan, Italy, August 1987.
- [CM99] Tony F. Chan and Pep Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM Journal on Applied Mathematics*, 36(2):354–367, February 1999.
- [CMS98] V. Caselles, J.-M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3):376–386, March 1998.
- [COL85] S. R. Curtis, A. V. Oppenheim, and J. S. Lim. Reconstruction of two-dimensional signals from threshold crossings. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 10, pages 1057–1060, Tampa, FL, March 1985.
- [CRP14] Y. Chen, R. Ranftl, and T. Pock. A bi-level view of inpainting – based image compression. In Z. Kúkelová and J. Heller, editors, *Proc. 19th Computer Vision Winter Workshop*, pages 61–66, Křtiny, Czech Republic, February 2014. Czech Pattern Recognition Society.
- [CS01a] T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (CDD). *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001.
- [CZ00] T. F. Chan and H. M. Zhou. Total variation improved wavelet thresholding in image compression. In *Proc. Seventh International Conference on Image Processing*, volume II, pages 391–394, Vancouver, Canada, September 2000.

- [DDI06] L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.
- [DFI02] Nira Dyn, Michael S Floater, and Armin Iske. Adaptive thinning for bivariate scattered data. *Journal of Computational and Applied Mathematics*, 145(2):505–517, 2002.
- [DFTT11] G. Di Blasi, E. Francomano, A. Tortorici, and E. Toscano. A smoothed particle image reconstruction method. *Calcolo*, 48(1):61–74, 2011.
- [Di 86] S. Di Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics and Image Processing*, 33:116–125, 1986.
- [DI04] Laurent Demaret and Armin Iske. Advances in digital image compression by adaptive thinning. *Annals of the MCFEA*, 3:105–109, 2004.
- [DMMH96] U. Y. Desai, M. M. Mizuki, I. Masaki, and B. K. P. Horn. Edge and mean based image compression. Technical Report 1584 (A.I. Memo), Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, U.S.A., November 1996.
- [DNV97] R. Distasi, M. Nappi, and S. Vitulano. Image compression by B-tree triangular coding. *IEEE Transactions on Communications*, 45(9):1095–1100, September 1997.
- [dR68] C. de Boor and J. R. Rice. Least squares cubic spline approximation II — variable knots. Technical Report CSD TR 21, Department of Computer Sciences, Purdue University, April 1968.
- [Dro93] L. Dron. The multiscale veto model: A two-stage analog network for edge detection and image reconstruction. *International Journal of Computer Vision*, 11(1):45–61, August 1993.
- [Duc76] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *RAIRO Mathematical Models and Methods in the Applied Sciences*, 10:5–12, 1976.

- [EBY99] G. Evans, J. Blackledge, and P. Yardley. *Numerical Methods for Partial Differential Equations*. Springer, Berlin, 1999.
- [Eld99] J. H. Elder. Are edges incomplete? *International Journal of Computer Vision*, 34(2/3):97–122, 1999.
- [ELPZ97] Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.
- [Fai90] J. Fairfield. Toboggan contrast enhancement for contrast segmentation. In *Proc. Tenth International Conference on Pattern Recognition*, volume 1, pages 712–716, Atlantic City, NJ, June 1990. IEEE Computer Society Press.
- [FKN73] S. Fučík, A. Kratochvíl, and J. Nečas. Kačanov–Galerkin method. *Commentationes Mathematicae Universitatis Carolinae*, 14(4):651–659, 1973.
- [FLA<sup>+</sup>06] G. Facciolo, F. Lecumberry, A. Almansa, A. Pardo, V. Caselles, and B. Rougé. Constrained anisotropic diffusion and some applications. In *Proc. 2006 British Machine Vision Conference*, volume 3, pages 1049–1058, Edinburgh, Scotland, September 2006.
- [For96] G. E. Ford. Application of inhomogeneous diffusion to image and video coding. In *Proc. 13th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 926–930, Asilomar, CA, November 1996.
- [FS76] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. In *Proc. Society of Information Display*, volume 17, pages 75–77, 1976.
- [FV62] D. S. Feingold and R. S. Varga. Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem. *Pacific Journal of Mathematics*, 12(4):1241–1250, 1962.
- [GG90] P. Grattoni and A. Guiducci. Contour coding for image description. *Pattern Recognition Letters*, 11(2):95–105, February 1990.

- [GMG12] Josselin Gautier, Olivier Le Meur, and Christine Guillemot. Efficient depth map compression based on lossless edge coding and diffusion. In *Picture Coding Symposium*, pages 81–84, Kraków, Poland, May 2012.
- [Gre13] S. Grewenig. *Fast Explicit Methods for PDE-based Image Analysis*. PhD thesis, Department of Mathematics, Saarland University, Germany, April 2013.
- [GWB10] S. Grewenig, J. Weickert, and A. Bruhn. From box filtering to fast explicit diffusion. In M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, editors, *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 533–542. Springer, Berlin, 2010.
- [Gwo12] P. Gwosdek. *Hardware-Accelerated Algorithms in Visual Computing*. PhD thesis, Department of Computer Science, Saarland University, Germany, July 2012.
- [GWSB13] S. Grewenig, J. Weickert, C. Schroers, and A. Bruhn. Cyclic schemes for pde-based image analysis. Technical Report 327, Dept. of Mathematics, Saarland University, Saarbrücken, Germany, March 2013.
- [GWW+05] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Towards PDE-based image compression. In N. Paragios, O. Faugeras, T. Chan, and C. Schnörr, editors, *Variational, Geometric and Level-Set Methods in Computer Vision*, volume 3752 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Berlin, 2005.
- [GWW+08] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2–3):255–269, July 2008.
- [GZG+10] P. Gwosdek, H. Zimmer, S. Grewenig, A. Bruhn, and J. Weickert. A highly efficient GPU implementation for variational optic flow based on the Euler-Lagrange framework. In *Proc. 3rd ECCV Workshop Computer Vision with GPUs*, Heraklion, Greece, September 2010. Springer, Berlin, Germany.



- [Hac85] W. Hackbusch. *Multigrid Methods and Applications*. Springer, New York, 1985.
- [Hig02] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.
- [HKM<sup>+</sup>98] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge. The emerging JBIG2 standard. *IEEE Trans. Circuits and Systems for Video Technology*, 8(7):838–848, November 1998.
- [HM89] R. Hummel and R. Moniot. Reconstructions from zero-crossings in scale space. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:2111–2130, 1989.
- [How10] M. Howison. Comparing gpu implementations of bilateral and anisotropic diffusion filters for 3d biomedical datasets. Technical Report LBNL-3425E, Lawrence Berkeley National Laboratory, Berkeley, CA, 2010.
- [HS81] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
- [HSW13] L. Hoeltgen, S. Setzer, and J. Weickert. An optimal control approach to find sparse data for laplace interpolation. In A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, volume 8081 of *Lecture Notes in Computer Science*, pages 151–164. Springer, Berlin, 2013.
- [Huf52] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.
- [Iij59] T. Iijima. Basic theory of pattern observation. In *Papers of Technical Group on Automata and Automatic Control*. IECE, Japan, December 1959. In Japanese.
- [Joi93] Joint Bi-level Image Experts Group. Information technology – progressive lossy/lossless coding of bi-level images. ISO/IEC JTC1 11544, ITU-T Rec. T.82, 1993. Final Committee Draft 11544.

- [Joi99] Joint Bi-level Image Experts Group. Information technology – lossy/lossless coding of bi-level images. ISO/IEC 14492, ITU-T Rec. T.88, 1999. Final Committee Draft 14492.
- [JSGA86] P. Johansen, S. Skelboe, K. Grue, and J. D. Andersen. Representing signals by their toppoints in scale space. In *Proc. Eighth International Conference on Pattern Recognition*, pages 215–217, Paris, France, October 1986.
- [KIK85] M. Kunt, A. Ikonomopoulos, and M. Kocher. Second-generation image-coding techniques. *Proceedings of the IEEE*, 73(4):549–574, April 1985.
- [KLD<sup>+</sup>05] F. M. W. Kanters, M. Lillholm, R. Duits, B. J. P. Jansen, B. Platel, L.M.J. Florack, and B. M. ter Haar Romeny. On image reconstruction from multiscale top points. In R. Kimmel, N. Sochen, and J. Weickert, editors, *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 431–439. Springer, Berlin, 2005.
- [KS05] I. Kopilovic and T. Szirányi. Artifact reduction with diffusion preprocessing for image compression. *Optical Engineering*, 44(2):1–14, February 2005.
- [KSFR07] H. Köstler, M. Stürmer, C. Freundl, and U. Rüde. PDE based video compression in real time. Technical Report 07-11, Lehrstuhl für Informatik 10, Univ. Erlangen–Nürnberg, Germany, 2007.
- [Kuh95] M. Kuhn. Effiziente Kompression von bi-level Bilddaten durch kontextsensitive arithmetische Codierung. Studienarbeit, Institut für Mathematische Maschinen und Datenverarbeitung der Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, July 1995.
- [KW93] H. Knutsson and C.-F. Westin. Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In *Proc. 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 515–523, New York, NY, 1993. IEEE Computer Society Press.

- [KZ96] R. Klette and P. Zamperoni. *Handbook of Image Processing Operators*. Wiley, New York, 1996.
- [Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.
- [LNG03] M. Lillholm, M. Nielsen, and L. D. Griffin. Feature-based image analysis. *International Journal of Computer Vision*, 52(2/3):73–95, 2003.
- [LSJO12] Yun Li, M. Sjostrom, U. Jennehag, and R. Olsson. A scalable coding approach for high quality depth image compression. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 1–4, October 2012.
- [LSW<sup>+</sup>07b] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. Image compression with edge-based inpainting. *IEEE Transactions on Circuits, Systems and Video Technology*, 17(10):1273–1286, October 2007.
- [Lun11] R. Lund. 3-D data compression with homogeneous diffusion. Master’s Thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany, 2011.
- [Mah05] M. Mahoney. Adaptive weighing of context models for lossless data compression. Technical Report CS-2005-16, Florida Institute of Technology, Melbourne, Florida, December 2005.
- [Max60] J. Max. Quantisation for minimum distortion. *IEEE Transactions on Information Theory*, 6:7–12, 1960.
- [MG80] A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations*. Wiley, Chichester, 1980.
- [MH80] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London, Series B*, 207:187–217, 1980.
- [MM98a] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 259–263, Chicago, IL, October 1998.

- [MM05] K. W. Morton and L. M. Mayers. *Numerical Solution of Partial Differential Equations*. Cambridge University Press, Cambridge, UK, second edition, 2005.
- [Mof90] A. Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921, November 1990.
- [MZ92a] S. Mallat and S. Zhong. Characterisation of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:720–732, 1992.
- [OBB<sup>+</sup>13] A. Orzan, A. Bousseau, P. Barla, H. Winnemöller, J. Thollot, and D. Salesin. Diffusion curves: a vector representation for smooth-shaded images. *Communications of the ACM*, 56(7):101–108, 2013.
- [OBW<sup>+</sup>08] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves: A vector representation for smooth-shaded images. *ACM Transactions on Graphics (Proc. SIGGRAPH '08)*, 27(3):92:1–8, 2008.
- [Pet12] P. Peter. Three-dimensional data compression with anisotropic diffusion. In *Proc. DAGM-OAGM 2012 Symposium for Pattern Recognition, Young Researchers Forum*, Berlin, 2012. Springer.
- [PM92] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Springer, New York, 1992.
- [PW14] P. Peter and J. Weickert. Colour image compression with anisotropic diffusion. In *Proc. 2014 IEEE International Conference on Image Processing*, Paris, France, October 2014. in press.
- [Ris76] J. J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.
- [RM07] A. Roussos and P. Maragos. Vector-valued image interpolation by an anisotropic diffusion-projection PDE. In F. Sgallari, F. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 104–115. Springer, Berlin, 2007.

- [RMB97] M. M. Reid, R. J. Millar, and N. D. Black. Second-generation image coding: An overview. *ACM Computing Surveys*, 29(1):3–29, 1997.
- [RSB03] S. D. Rane, G. Sapiro, and M. Bertalmio. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. *IEEE Transactions on Image Processing*, 12(3):296–302, March 2003.
- [Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, second edition, 2003.
- [Sch97c] H. R. Schwarz. *Numerische Mathematik*. Teubner, Stuttgart, fourth edition, 1997.
- [Sch12] C. Schmaltz. *Compression, Pose Tracking, and Halftoning*. PhD thesis, Department of Computer Science, Saarland University, Germany, March 2012.
- [SCSA04] A. Solé, V. Caselles, G. Sapiro, and F. Arandiga. Morse description and geometric encoding of digital elevation maps. *IEEE Transactions on Image Processing*, 13(9):1245–1262, September 2004.
- [SGBW10] C. Schmaltz, P. Gwosdek, A. Bruhn, and J. Weickert. Electrostatic halftoning. *Computer Graphics Forum*, 29(8):2313–2327, December 2010.
- [SH93] C. Saloma and P. Haerberli. Two-dimensional image reconstruction from Fourier coefficients computed directly from zero crossings. *Applied Optics*, 32(17):3092–3093, April 1993.
- [SSW14] C. Schroers, S. Setzer, and J. Weickert. A variational taxonomy for surface reconstruction from oriented points. *Computer Graphics Forum (Proc. EUROGRAPHICS 2014)*, 2014.
- [SW12] C. Schmaltz and J. Weickert. Video compression with 3-d pose tracking, pde-based image coding, and electrostatic halftoning. In A. Pinz, T. Pock, H. Bischof, and F. Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 438–447. Springer, 2012.

- [SWB09] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition, Lecture Notes in Computer Science*, pages 452–461, Berlin, 2009. Springer.
- [SWL06] Xiaoyan Sun, Feng Wu, and Shipeng Li. Compression with vision technologies. In *Picture Coding Symposium*, pages 1–5, Beijing, China, April 2006.
- [Tan10] C. H. Tang. Using the (1+1) EA for optimizing image interpolation with homogenous diffusion. Master’s Thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany, 2010.
- [TD05] D. Tschumperlé and R. Deriche. Vector-valued image regularization with PDEs: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):506–516, April 2005.
- [TM02] D. S. Taubman and M. W. Marcellin, editors. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer, Boston, 2002.
- [TOS01] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, San Diego, 2001.
- [TS06] B. Triggs and M. Sdika. Boundary conditions for Young - van Vliet recursive filtering. *IEEE Transactions on Signal Processing*, 54(5):1–2, May 2006.
- [TSYK02] H. Tsuji, T. Sakatani, Y. Yashima, and N. Kobayashi. A nonlinear spatio-temporal diffusion and its application to prefiltering in MPEG-4 video coding. In *Proc. 2002 IEEE International Conference on Image Processing*, volume 1, pages 85–88, Rochester, NY, September 2002.
- [US05] K. Uhler and V. Skala. Reconstruction of damaged images using radial basis functions. In *Proc. 13th European Signal Processing Conference (EUSIPCO)*, pages 160–163, Antalya, Turkey, September 2005.
- [WB02] J. Weickert and T. Brox. Diffusion and regularization of vector- and matrix-valued images. In M. Z. Nashed and

- O. Scherzer, editors, *Inverse Problems, Image Analysis, and Medical Imaging*, volume 313 of *Contemporary Mathematics*, pages 251–268. AMS, Providence, 2002.
- [Wei96b] J. Weickert. Foundations and applications of nonlinear anisotropic diffusion filtering. *Zeitschrift für angewandte Mathematik und Mechanik*, 76(Suppl. 1):283–286, 1996.
- [Wei96e] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing Supplement*, 11:221–236, 1996.
- [Wei98a] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart, 1998.
- [Wei99c] J. Weickert. Nonlinear diffusion filtering. In B. Jähne, H. Haußecker, and P. Geißler, editors, *Handbook on Computer Vision and Applications, Vol. 2: Signal Processing and Pattern Recognition*, pages 423–450. Academic Press, San Diego, 1999.
- [Wer35] H. Werner. Studies on contour. *The American Journal of Psychology*, 47(1):40–64, January 1935.
- [Wes04] P. Wesseling. *An Introduction to Multigrid Methods*. R. T. Edwards, Flourtown, 2004.
- [Wit83] A. P. Witkin. Scale-space filtering. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, volume 2, pages 945–951, Karlsruhe, West Germany, August 1983.
- [WSWX06] C. Wang, X. Sun, F. Wu, and H. Xiong. Image compression with structure-aware inpainting. In *Proc. IEEE International Symposium on Circuits and Systems*, pages 1816–1819. IEEE Computer Society Press, May 2006.
- [WW06] J. Weickert and M. Welk. Tensor field interpolation with PDEs. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields*, pages 315–325. Springer, Berlin, 2006.
- [WWW13] J. Weickert, M. Welk, and M. Wickert. L2-stable non-standard finite differences for anisotropic diffusion. In

- A. Kuijper, T. Pock, K. Bredies, and H. Bischof, editors, *Scale-Space and Variational Methods in Computer Vision*, volume 7893 of *Lecture Notes in Computer Science*, pages 380–391. Springer, Berlin, June 2013.
- [WZSG09] Y. Wu, H. Zhang, Y. Sun, and H. Guo. Two image compression schemes based on image inpainting. In *Proc. 2009 International Joint Conference on Computational Sciences and Optimization*, pages 816–820. IEEE Computer Society Press, April 2009.
- [XFC<sup>+</sup>07] Z. Xie, W. R. Franklin, B. Cutler, M. A. Andrade, M. Inanc, and D. M. Tracy. Surface compression using over-determined Laplacian approximation. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations XVII*, volume 6697 of *Proceedings of SPIE*. SPIE Press, Bellingham, 2007.
- [XSWL07] Z. W. Xiong, X. Y. Sun, F. Wu, and S. P. Li. Image coding with parameter-assistant inpainting. In *Proc. 2007 IEEE International Conference on Image Processing*, volume 2, pages 369–372, San Antonio, TX, September 2007.
- [You71] D. M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.
- [YP86] A. L. Yuille and T. A. Poggio. Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):15–25, January 1986.
- [ZR86] Y. Zeevi and D. Rotem. Image reconstruction from zero-crossings. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34:1269–1277, 1986.