Efficient Methods for Inpainting-Based Image Compression

A dissertation submitted towards the degree
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by
Rahul Mohideen KAJA MOHIDEEN

Saarbrücken, 2024

Day of Colloquium:
09.12.2024

Dean of Faculty:
Prof. Dr. Roland Speicher

Chair of the Committee:
Prof. Dr. Dietrich Klakow

Reviewers:
Prof. Dr. Joachim Weickert
Prof. Dr. Irena Galić

Academic Assistant:
Dr. Anna Kukleva

"Every now and then, a man's mind is stretched by a new idea or sensation and never shrinks back to its former dimensions."

Oliver Wendell Holmes Sr.

Short Abstract

This thesis is dedicated to image compression methods that optimise and store a small subset of the image pixels, called the mask, and reconstruct the rest of the image through inpainting. Inpainting is the general term for filling in missing or damaged parts of an image. However, saving fully optimised pixel positions is expensive and has, till now, received little attention. We propose two new families of codecs specifically addressing this problem that offer better performance or a better trade-off between speed and performance. We achieve this by extensively evaluating existing image and data compression methods, understanding their most successful principles and combining them to form new codecs. Inpainting-based image compression methods have long since employed partial differential equations for their inpainting method. They offer good reconstruction quality in most cases. However, naive implementations are very slow, requiring significant efforts to be sped up. In this thesis, we explore Shepard inpainting as a simple and efficient alternative. By considering data selection and extensions such as anisotropy, we propose our own Shepard-inpainting-based compression methods that offer a good mix of simplicity, efficiency, and quality. Multi-channel images are encountered more than single-channel images today. However, very few inpainting-based codecs have dedicated colour modes. We propose colour extensions for Shepard inpainting, including a luma-preference mode and a vector quantisation mode that has not yet been proposed. The resulting codecs offer better compression performance than the base RGB mode on colour images. This thesis tries to present efficient and better methods to compress fully optimised masks and to see how far one can go with a simple but efficient operator such as Shepard inpainting.

Kurzzusammenfassung

Diese Dissertation beschäftigt sich mit Bildkompressionsmethoden, die eine kleine Teilmenge der optimierten Bildpixel, die sogenannte Maske, speichern und den Rest des Bildes durch Inpainting rekonstruieren. Inpainting ist der allgemeine Begriff für die Rekonstruktion fehlender oder beschädigter Teile eines Bildes. Das Speichern vollständig optimierter Pixelpositionen ist jedoch teuer und wurde bisher wenig beachtet. Wir führen zwei neue Codec-Familien ein, die sich speziell diesem Problem widmen und eine bessere Leistung oder einen besseren Kompromiss zwischen Geschwindigkeit und Leistung bieten. Wir erreichen dies, indem wir bestehende Bild- und Datenkompressionsmethoden umfassend evaluieren und ihre erfolgreichsten Prinzipien kombinieren, um neue Methoden zu entwickeln. Inpainting-basierte Bildkompressionsmethoden verwenden seit langem partielle Differentialgleichungen für das Inpainting. Sie bieten in den meisten Fällen eine gute Rekonstruktionsqualität. Naive Implementierungen sind jedoch sehr langsam und erfordern erhebliche Anstrengungen, um sie zu beschleunigen. Diese Arbeit untersucht Shepard-Inpainting als einfache und effiziente Alternative. Unter Berücksichtigung der Datenauswahl und Erweiterungen wie Anisotropie führen wir unsere eigenen, auf Shepard-Inpainting basierenden Kompressionsmethoden ein, die eine gute Mischung aus Einfachheit, Effizienz und Qualität bieten. Nur wenige auf Inpainting basierende Codecs verfügen über dedizierte Farbmodi, obwohl Farbbilder beliebter sind. Wir führen Farbmodi für Shepard-Inpainting ein, darunter einen Luma-Präferenzmodus und einen Modus mit Vektorquantisierung. Die daraus resultierenden Methoden bieten bei Farbbildern eine verbesserte Kompressionsleistung gegenüber dem Standardverfahren. Diese Dissertation versucht, effiziente und bessere Methoden zum Komprimieren vollständig optimierter Masken vorzustellen und zu sehen, wie weit man mit einem einfachen, aber effizienten Inpainting-Operator wie Shepard-Inpainting kommen kann.

Abstract

Inpainting-based codecs (coders and decoders) store sparse, quantised pixel data and decode by reconstructing the discarded image parts. This process of filling in the missing regions of the image just from the stored pixel information is called inpainting. Storing the carefully optimised positions of known data creates a lossless compression problem on sparse and often scattered binary images. This central issue is crucial for the performance of such codecs. Since it has only received little attention in the literature, we have conducted the first systematic investigation of this problem. To this end, we first review and compare a wide range of existing methods, from image compression and general-purpose coding, regarding their coding efficiency and runtime. Afterwards, an ablation study enables us to identify and isolate the most valuable components of existing methods. We combine those ingredients into new codecs that offer better compression ratios or a more favourable trade-off between speed and performance.

Successful inpainting-based image compression codecs traditionally use inpainting operators that solve partial differential equations. This requires some numerical expertise if efficient implementations are necessary. Our goal is to investigate variants of Shepard inpainting as simple alternatives for inpainting-based compression. They can be implemented efficiently when we localise their weighting function. To turn them into viable codecs, we have to introduce novel extensions of classical Shepard interpolation that adapt successful ideas from previous codecs: Anisotropy allows direction-dependent inpainting, which improves reconstruction quality. Additionally, we incorporate data selection by subdivision as an efficient way to tailor the stored information to the image structure. On the encoding side, we introduce the novel concept of joint inpainting and prediction for isotropic Shepard codecs, where storage cost can be reduced based on intermediate inpainting results. In an ablation study, we show the usefulness of these individual contributions and demonstrate that they offer synergies which elevate the performance of Shepard inpainting to surprising levels. Our resulting approaches offer a more favourable trade-off between simplicity and quality than traditional inpainting-based codecs. Experiments show that they can outperform JPEG and JPEG2000 at high compression ratios.

Few inpainting-based compression methods have a dedicated mode to compress colour images, even though multi-channel images are more prevalent than grayscale images. Therefore, we evaluate different approaches for inpainting-based colour compression focused on Shepard inpainting. However, the principles can be extended to other inpainting operators. Inpainting operators can reconstruct an extensive range of colours from a small colour palette of the known pixels. We exploit this with a luma preference mode, which uses higher sparsity in YCbCr colour channels than in the brightness channel. Furthermore, we propose the first full vector quantisation mode for an inpainting-based codec that stores only a small codebook of colours. Our experiments reveal that both colour extensions yield significant improvements.

This thesis is dedicated to making inpainting-based image compression more efficient. We aim for compression efficiency by targeting known pixel positions as they are more expensive to store and operational efficiency by using Shepard inpainting and improving it with our proposed extensions.

Zusammenfassung

Inpainting-basierte Codecs (Coder und Decoder) speichern spärliche, quantisierte Pixeldaten und decodieren, indem sie die verworfenen Pixeldaten rekonstruieren. Dieser Prozess der Rekonstruktion der fehlenden Bildbereiche nur aus den gespeicherten Pixelinformationen wird als Inpainting bezeichnet. Das Speichern der sorgfältig optimierten Positionen bekannter Daten führt zu einem verlustfreien Kompressionsproblem bei spärlichen und im Bild verstreuten binären Pixeldaten. Dieses zentrale Problem ist entscheidend für die Leistung solcher Codecs. Da es in der Literatur nur wenig Beachtung gefunden hat, haben wir die erste systematische Untersuchung dieses Problems durchgeführt. Zu diesem Zweck überprüfen und vergleichen wir zunächst eine breite Palette bestehender Methoden, von der Bildkompression bis zur Allzweckcodierung, hinsichtlich ihrer Codierungseffizienz und Laufzeit. Anschließend ermöglicht uns eine Ablationsstudie, die zweckvollsten Komponenten bestehender Methoden zu identifizieren und zu isolieren. Wir kombinieren diese Bestandteile zu neuen Codecs, die bessere Kompressionsverhältnisse oder einen günstigeren Kompromiss zwischen Laufzeit und Leistung bieten.

Erfolgreiche inpainting-basierte Bildkompressionscodecs verwenden traditionell Inpainting-Operatoren, die partielle Differentialgleichungen lösen. Dies erfordert einige numerische Fachkenntnisse, wenn effiziente Implementierungen erforderlich sind. Unser Ziel ist es, Varianten des Shepard-Inpaintings als einfache Alternativen für inpainting-basierte Kompression zu untersuchen. Sie können effizient implementiert werden, wenn wir ihre Gewichtungsfunktion lokalisieren. Um sie in brauchbare Codecs umzuwandeln, müssen wir neuartige Erweiterungen der klassischen Shepard-Interpolation einführen, die erfolgreiche Ideen aus früheren Codecs übernehmen: Anisotropie erlaubt richtungsabhängiges Inpainting, was die Rekonstruktionsqualität verbessert. Darüber hinaus integrieren wir die Datenauswahl durch Bildunterteilung als effiziente Möglichkeit, die gespeicherten Informationen an die Bildstruktur anzupassen. Auf der Kodierungsseite führen wir das neuartige Konzept der gemeinsamen Vorhersage und Rekonstruktion für isotrope Shepard-Codecs ein, bei denen die Speicherkosten basierend auf Zwischenergebnissen des Inpaintings gesenkt werden können. In einer Ablationsstudie zeigen wir die Nützlichkeit dieser einzelnen Beiträge und demonstrieren, dass sie Synergien bieten, die die Leistung von Shepard-Inpainting auf ein überraschendes Niveau heben. Unsere daraus resultierenden Ansätze bieten einen besseren Kompromiss zwischen Einfachheit und Qualität als herkömmliche inpainting-basierte Codecs. Unsere Experimente zeigen, dass sie JPEG und JPEG2000 bei hohen Kompressionsraten übertreffen können.

Nur wenige inpainting-basierte Kompressionsmethoden verfügen über einen dedizierten Modus zum Komprimieren von Farbbildern, obwohl Farbbilder häufiger vorkommen als Schwarzweißbilder. Daher bewerten wir verschiedene Ansätze für inpainting-basierte Farbkompression mit Schwerpunkt auf Shepard-Inpainting. Die Prinzipien können jedoch auf andere Inpainting-Operatoren erweitert werden. Inpainting-Operatoren können aus einer kleinen Farbpalette der bekannten Pixel eine umfangreiche Farbpalette rekonstruieren. Wir nutzen dies für einem Luma-Präferenzmodus, der in den YCbCr-Farbkanälen weniger Daten verwendet als im Helligkeitskanal. Darüber hinaus führen wir den ersten vollständigen Vektorquantisierungsmodus für einen inpainting-basierten Codec ein, der nur eine kleine Farbpalette speichert. Unsere Experimente zeigen, dass beide Erweiterungen zu erheblichen Verbesserungen führen.

Diese Dissertation widmet sich der effizienteren inpainting-basierten Bildkompression. Unser Ziel ist es, teure bekannte Pixelpositionen besser zu komprimieren und eine höhere Leistung zu erreichen, indem wir Shepard-Inpainting verwenden und es mit unseren vorgeschlagenen Erweiterungen verbessern.

Acknowledgements

Firstly, I would like to express my utmost thanks to my advisor *Prof. Dr Joachim Weickert*, whose wisdom and mentorship were an essential part of my work in the Mathematical Image Analysis group.

I am deeply grateful to *Dr Pascal Peter*, without whom this thesis would not have been possible. He has been with me every step of the way since I was a student. His ideas and suggestions are prevalent in every facet of all my published work and this thesis. His work on inpainting-based image compression using Shepard interpolation is the basis of a significant portion of this thesis. I would also like to thank him for proofreading this thesis manuscript and offering me his feedback, which was vital to its completion.

I want to acknowledge the *European Research Council (ERC)* for their funding under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 741215, ERC Advanced Grant INCOVID).

Next, I would like to thank my co-authors from the Mathematical Image Analysis group who contributed to the scientific material of my work: *Dr Tobias Alt* for his work on anisotropic Shepard inpainting and the direct tonal optimisation for isotropic Shepard inpainting; *Sarah Andris* for her expertise in inpainting-based video compression; *Vassillen Chizhov* for providing his implementation of Voronoi decomposition, which was integrated into the subdivision-based anisotropic Shepard codec; *Dr Matthias Augustin* for his insights on numerical methods and for providing the Green's functions implementation of tonal optimisation for homogeneous diffusion, which was used for comparisons in Section 4.4.1; and Alexander Scheer for his work on vector quantisation. I am also indebted to the other members of my group, current and former, for the engaging and fruitful discussions and for creating a pleasant work environment.

I would finally like to thank my family: *Anesha Banu, Kaja, and Nisha Mohideen, Darko, Damira, and David Sklezur* for their unconditional support and my fiancé *Rebecca Sklezur* for standing by my side all this time.

Contents

Chapter 1

Introduction

Inpainting, as the name suggests, is an image restoration technique that was introduced to reconstruct damaged or missing regions of an image, physical or digital [20]. Masnou and Morel [88] proposed an early inpainting model based on variational methods. The use of partial differential equations (PDEs) for inpainting was proposed by Bertalmío et al. [20]. Inpainting for textures was then proposed by Efros and Leung [43], where they used similar regions in other parts of the image to inpaint, also known as exemplar-based inpainting.

Currently, our lives revolve almost exclusively around digital media. We take pictures and videos with our mobile phones and upload them, download images or stream videos. Today's average mobile phone can capture images at a resolution of 20 MP ($20 \cdot 10^6$ pixels), and some phones can even capture at a resolution of 100 MP. Assuming an RGB image with three channels and each channel is stored with a bit-depth of 8, that amounts to approximately 60 MB ($60 \cdot 10^6$) for a 20 MP image. Even though storage is very cheap today, storing an image as is is still inefficient, especially when it will most likely be transferred somewhere else via the internet. The inefficiency of storing data in its original form is much more evident in the case of videos. A full HD video of five seconds with a resolution of 1920 pixels by 1080 pixels at 24 frames per second amounts to 746 MB ($746 \cdot 10^6$), whereas a 4K video of the same length and framerate would need four times as much storage. With our current internet speeds, video streaming would be utterly infeasible if the videos were downloaded as is. However, transferring images or streaming videos quickly is currently possible, mainly because we compress them before sending them.

Data compression is the process of compressing a source data stream such that it occupies less storage space. This can be done by rearranging and reformulating the source data where no information is discarded, also called lossless compression. This process is completely reversible and is performed primarily on data we would like to keep unaltered, for example, text files. However, higher compression performance can be obtained by irreversibly eliminating some data. This is performed mainly on images and video by throwing away perceptually unimportant data. Therefore, they will look very similar to the original image or video to the human eye while taking up much less storage space.

In addition to its primary purpose as a restoration method for damaged images, inpainting has found a new purpose as a method to compress images and videos. Inpainting-based image compression is unconventional in two aspects: It drives inpainting to the extreme by considering very sparse data and combines it with data optimisation. In the encoding step, one stores only a tiny, carefully optimised fraction of the image pixels. During the decoding phase, the unknown data are approximated by inpainting. Since inpainting-like filling-in mechanisms are postulated to play an essential role in the human visual system [134], inpainting-based compression appears natural and conceptually appealing. Moreover, aiming at sparsity in the spatial domain is particularly simple and distinguishes inpainting-based compression from widely-used transform-based approaches such as JPEG [97], JPEG2000 [121], and HEVC intra [44]. The latter ones aim at sparsity in the discrete cosine or wavelet domain, which is achieved by applying a transform to the target domain and quantising the coefficients coarsely. Advanced inpainting-based codecs can outperform JPEG2000 [111], and they can be far ahead of the state-of-the-art for data with a low to moderate amount of texture, such as depth maps [66].

Inpainting-based image compression methods store a fraction of the image pixels and reconstruct the rest during decoding. There are two aspects to optimise when selecting the pixels to store. One can optimise the positions of the stored pixels *(spatial optimisation)* as well as their corresponding grey or colour values *(tonal optimisation)*. The positions of the stored pixels constitute the *inpainting mask*. While numerous methods have been proposed for spatial and tonal optimisation, there is a general tradeoff between simplicity, efficiency, and quality.

However, these approaches require careful optimisation of both the positions and values of the known data. To obtain maximum reconstruction quality, mask optimisation methods should be allowed to freely choose any pixel as a mask point [19, 31, 37, 55, 70, 85]. This is called unconstrained optimisation. This creates a delicate compression problem: Generally, storing optimised data is expensive. Any deviations, e.g. by inexact, cheaper positions, can also reduce the reconstruction quality during decompression. While having deviations in pixel values is feasible, and many inpainting operators are robust under quantisation [100, 102, 111], storing pixel locations is a much more sensitive task. Storing pixel locations is equivalent to storing a sparse binary image.

Most inpainting operators perform best for unconstrained, optimised data stored losslessly [102]. Consequently, many optimisation methods [19, 31, 37, 55, 70, 85] would benefit from codecs specifically aimed at optimised inpainting masks. Even beyond inpainting-based compression, unconstrained mask codecs could be useful for other applications such as storage of sparse image features (e.g. SIFT [78], SURF [17]). However, there is only a small number of publications [38, 87, 102] that directly address the compression of unconstrained masks. In Chapter 3, we will implement a new family of codecs specifically tailored for the compression of sparse binary images and compare them to existing general-purpose image compression methods and other specialised methods.

Apart from a few notable exceptions such as exemplar-based inpainting [70], inpainting with concepts from Smoothed Particle Hydrodynamics [36], and linear spline inpainting [37, 41, 87], most inpainting-based codecs (coders and decoders) employ partial differential equations (PDEs) of diffusion type for inpainting. Homogeneous diffusion allows very efficient algorithms if one uses sophisticated numerical ideas [32, 59, 68, 69, 72, 84] while edge-enhancing anisotropic diffusion offers the highest quality due to its anisotropy [48, 111].

On the other hand, Shepard interpolation relies on the idea of normalised weighted averaging. If one uses a localised weighting function, only a few surrounding mask pixels influence a given unknown pixel. The locality of the resulting Shepard inpainting allows simple and fast inpainting and tonal optimisation. This distinguishes it from PDE-based methods, where high efficiency in the inpainting step is possible but requires the adaptation of advanced numerical concepts such as multigrid techniques [72, 84], Fast Explicit Diffusion [104], Green's functions [59, 68], finite element methods [32], and domain decomposition approaches [69]. Moreover, all exact methods for the tonal optimisation of PDE-based approaches are relatively time-consuming and may be substantially slower than a fast PDE-based inpainting step. Shepard interpolation is non-iterative and requires less numerical expertise. The original paper by Shepard [116] intended this method to be used for interpolation, whereas we use a variant proposed by Achanta et al. [2], which performs approximation instead. We prefer this version since its ability to modify known data can be helpful for our compression purposes. Shepard inpainting provides a solid alternative to PDE-based inpainting that offers a better compromise than existing approaches w.r.t. implementational simplicity, computational efficiency, and approximation quality. In Chapter 4, We explore this further and its viability and performance in an image compression setting by proposing a family of simple and highly efficient end-to-end codecs that do not require the numerical sophistication of PDE-based codecs while keeping certain quality-critical features such as anisotropy [5] and inpainting mask optimisation.

Although inpainting-based image compression methods are mainly proposed and developed for single-channel images, multi-channel or colour images make up most of the images we see daily. Contemporary transform-based codecs have dedicated colour modes to compress colour images. For instance, JPEG [97] and JPEG2000 [121] use chroma subsampling in YCbCr space, guided by the idea that structural information is visually more important than colour. Discarding some of the information in the colour channels Cb and Cr allows them to store more accurate transform coefficients for the structure component of the image. On colour images, inpainting-based codecs rely on coarse quantisation of RGB values. Interestingly, inpainting operators tend to fill in gaps not only in the spatial domain but also in the co-domain of colour values. Despite this property, little research has been invested into colour modes for inpainting-based codecs. In particular, the potential of techniques that create sparse colour palettes, such as vector quantisation, remains unexplored. In Chapter 5, we will look at multi-channel specific concepts such as vector quantisation and luma preference and apply them to inpainting-based compression methods, specifically codecs which use Shepard inpainting.

Chapter 2

Related Work

Every inpainting-based image compression method has three vital components: the inpainting operator, data selection strategy (spatial and tonal), and encoding of the selected data. In this section, we dive further into previous work done on each of these three components.

2.1 Inpainting Operators

2.1.1 Overview

Since the inpainting operator recovers the missing image parts from the known data, it is crucial for the reconstruction quality. A significant number of operators use partial differential equations (PDEs). In particular, homogeneous diffusion [63] is a popular choice for compression [27, 49, 58, 65, 84, 102] since it is simple and fast compared to other PDE-based methods. It is an isotropic operator, i.e. it propagates information from stored pixels equally in all directions. As a higher-order alternative to homogeneous diffusion, biharmonic inpainting [42] is another applicable isotropic operator for sparse image inpainting [31, 48, 111]. Last but not least, anisotropic variants of nonlinear PDEs have been explored, most notably edge-enhancing diffusion (EED) [48, 130, 132] and higher-order variants [67, 103]. They adapt themselves to the local data structure. EED is the core component in some of the qualitatively best diffusion-based compression methods such as R-EED [111] and R-EED-LP [103].

While diffusion-based inpainting methods perform very well on piecewise smooth and mildly textured images, they struggle with high-frequent texture data. To address this particular issue, sparse exemplar-based inpainting methods have been proposed [45]. They reconstruct images by copying pixels or whole image patches from similar neighbourhoods. It is also possible to combine diffusion and exemplar-based inpainting methods in hybrid codecs [106].

Deep learning-based inpainting methods have also generated interest [96, 137, 138]. Successful concepts include generative adversarial networks (GANs) [99, 125], deep prior approaches [124], and stable diffusion [79]. While deep learning approaches are undoubtedly powerful, they are computationally expensive to train, and the models are not as transparent as PDE-based inpainting.

Shepard interpolation [116] is a simple and straightforward inpainting operator. It can be interpreted as a special case of a class of inpainting operators known as radial basis functions (RBFs) [25, 53] which have been successfully applied for scattered data interpolation [12, 40, 80, 133]. Also, anisotropic variations of RBFs have been considered [18, 28, 36]. The work of Daropoulos et al. [36] is closest in spirit to our work, as they employ anisotropic RBF kernels with spatial and tonal optimisation.

Multiple publications have proposed improvements for isotropic Shepard interpolation. This includes restrictions of Shepard interpolation to a localised averaging of known data [46], which is also crucial for our own applications. This influence area of known data has also been adapted locally [75, 107]. Shepard interpolation has been used successfully in sparse image inpainting [2, 71]. However, our conference publication [100] is the first that applies it to compression.

2.1.2 Inpainting with Diffusion Processes

Diffusion has a long tradition in image processing [63, 98, 129], and it has also been used for image inpainting; see e.g. [27, 132]. Let $f : \Omega \to \mathbb{R}$ denote a grey value image on a rectangular image domain $\Omega \subset \mathbb{R}^2$ that is only known on a subset $K \subset \Omega$, also called the *inpainting mask*. To reconstruct the unknown image data in $\Omega \setminus K$, diffusion-based inpainting computes the steady state ($t \to \infty$) of the following initial value problem:

$$\partial_t u = \operatorname{div}\left(\boldsymbol{D}\boldsymbol{\nabla} u\right) \qquad \text{on } \Omega \setminus K \times (0,\infty), \tag{2.1}$$

$$u\left(x,y,t\right) = f\left(x,y,0\right) \qquad \text{on } K \times [0,\infty), \tag{2.2}$$

$$\boldsymbol{n}^\top \boldsymbol{D}\boldsymbol{\nabla} u = 0 \qquad \text{on } \partial\Omega \times (0,\infty)\,. \tag{2.3}$$

Here, $u(x,y,t)$ denotes the image pixel value at position (x,y) and time t, and $\boldsymbol{n}$ is the outer normal vector at the image boundary $\partial\Omega$. The spatial gradient operator is denoted by $\boldsymbol{\nabla} = (\partial_x, \partial_y)^\top$, and div $= \boldsymbol{\nabla}^\top$ is the divergence. The diffusion tensor $\boldsymbol{D} \in \mathbb{R}^{2\times 2}$ is a positive semi-definite matrix. Its eigenvectors determine the propagation directions of the diffusion process, and their eigenvalues determine the amount of diffusion along those directions. The Dirichlet boundary conditions in Eq. (2.2) specify that known pixel values stay unmodified. Reflecting boundary conditions are defined in Eq. (2.3) to avoid diffusion across the image boundaries.

The simplest choice for the diffusion tensor is $\boldsymbol{D} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. In that case, we can write Eq. (2.1) as

$$\partial_t u = \operatorname{div}\left(\boldsymbol{\nabla} u\right) = \Delta u = \partial_{xx} u + \partial_{yy} u. \tag{2.4}$$

The above equation describes *homogeneous diffusion*, which propagates information isotropically in all directions [63].

There are more sophisticated choices for $\boldsymbol{D}$, which allow, e.g. direction-dependent (anisotropic) adaptation of the inpainting. For example, *edge-enhancing anisotropic diffusion* (EED) [130, 132] considers the diffusion tensor $\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)$, where u_σ represents the convolution of the evolving image u with a Gaussian kernel of standard deviation σ. This Gaussian convolution makes the edge detector $|\boldsymbol{\nabla} u_\sigma|^2$ more robust under noise, where $|\,.\,|$ denotes the Euclidean norm. The first normalised eigenvector of $\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)$ is chosen as $\boldsymbol{v}_1 = \boldsymbol{\nabla} u_\sigma / |\boldsymbol{\nabla} u_\sigma|$. It is perpendicular to the edge, while the second normalised eigenvector $\boldsymbol{v}_2$ is parallel to the edge. The eigenvalues μ_1 and μ_2 denote the contrast in the direction of these eigenvectors. By setting $\mu_2 = 1$, one allows full diffusion along edges. To reduce diffusion across edges, one uses for μ_1 a decreasing diffusivity such as the one by Charbonnier et al. [30]:

$$\mu_1 = g\left(|\boldsymbol{\nabla} u_\sigma|^2\right) = \frac{1}{\sqrt{1+\frac{|\boldsymbol{\nabla} u_\sigma|^2}{\lambda^2}}}. \tag{2.5}$$

with some contrast parameter $\lambda > 0$. With these choices, $\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)$ can be written as

$$\boldsymbol{D}\left(\boldsymbol{\nabla} u_\sigma\right) = g\left(|\boldsymbol{\nabla} u_\sigma|^2\right) \boldsymbol{v}_1 \boldsymbol{v}_1^\top + \boldsymbol{v}_2 \boldsymbol{v}_2^\top. \tag{2.6}$$

For inpainting with homogeneous diffusion or EED, one observes global convergence where the steady state does not depend on the initialisation. However, since a good initialisation can accelerate the convergence, a pragmatic approach is to initialise the non-mask pixels with the average grey value of the mask pixels.

For EED inpainting, one discretises the parabolic PDE (2.1) with finite differences and computes the reconstruction by means of numerical solvers [92]. An explicit time discretisation is simple but has to obey severe time step size restrictions for stability reasons. There are ways to accelerate explicit schemes by using cyclically varying time step sizes [131], or extrapolation ideas [52, 123]. A semi-implicit time discretisation does not suffer from any time step size limits [129] but requires solving a large linear system with a matrix that is symmetric, positive definite, and sparse. To this end, one can use iterative solvers such as conjugate gradients.

For homogeneous diffusion inpainting, efficient numerical solvers often exploit direct discretisations of the Laplace equation $\Delta u = 0$ that arises in the steady state. This has been done with multigrid methods [72, 84], discrete Green's function approaches [59, 68], finite element discretisations with conjugate gradient solvers [32], and domain decomposition algorithms [69].

These discussions show that diffusion-based inpainting requires quite some numerical expertise if one aims at highly efficient algorithms. This motivates us to study alternatives that also offer efficient algorithms but do not rely on such knowledge and lead to relatively simple implementations.

2.1.3 Inpainting with Radial Basis Functions

In contrast to diffusion-based approaches, inpainting with radial basis functions [25, 133] does not rely on a PDE-based formulation. As in the previous subsection, we want to reconstruct an image $\boldsymbol{u}$ from the known data $\boldsymbol{f} \in \mathbb{R}^{m \times n}$ known only on a subset K of the image domain Ω. An unknown pixel $u_i := u(\boldsymbol{x}_i)$ at $\boldsymbol{x}_i = (x_{1i}, x_{2i}) \in \Omega$ is reconstructed with the RBF w according to [53]

$$u(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in K} w(\boldsymbol{x}_j - \boldsymbol{x}_i) c_j \,. \tag{2.7}$$

A common choice for w are multiquadrics [133]. The coefficients c_j are determined by the interpolation condition

$$u(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) \quad \forall\, \boldsymbol{x}_j \in K \,. \tag{2.8}$$

Plugging these conditions into Eq. (2.7) yields a linear system of equations that can be solved for the coefficients c_j.

At first glance, RBF approaches seem to have little in common with the PDE-based inpainting from Section 2.1.2. However, Augustin et al. [12] have shown that linear diffusion inpainting and RBF interpolation can be expressed in a unifying pseudo-differential framework. Since we aim to find simple alternatives to diffusion-based inpainting for compression, we consider a simplified special case of RBF interpolation.

2.1.4 Isotropic Shepard Inpainting

Classical Shepard interpolation [116] computes the reconstructed values u_i directly from the known data f_i according to

$$u(\boldsymbol{x}_i) = \begin{cases} \frac{\sum_{\boldsymbol{x}_j \in K} w(\boldsymbol{x}_j - \boldsymbol{x}_i) f_j}{\sum_{\boldsymbol{x}_j \in K} w(\boldsymbol{x}_j - \boldsymbol{x}_i)}, & \text{if} \quad \forall\, j : |\boldsymbol{x}_i - \boldsymbol{x}_j| > 0 \,, \\ f_i, & \text{if} \quad \exists\, j : |\boldsymbol{x}_i - \boldsymbol{x}_j| = 0 \,. \end{cases} \tag{2.9}$$

In this approach, the family of weighting functions w is defined by

$$w\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right) = \frac{1}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^p} \,. \tag{2.10}$$

The exponent $p > 1$ controls the influence of neighbouring points, where higher values of p result in fewer contributions from distant points. Compared to the RBF formulation in Eq. (2.7) Shepard interpolation Eq. (2.9) explicitly includes the interpolation condition and does not require computing the coefficients c_j. Instead, it specifies them directly as

$$c_j := \frac{f_j}{\sum_{\boldsymbol{x}_j \in K} w\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right)}, \tag{2.11}$$

considerably simplifying the interpolation problem.

We can further simplify the computation and reduce the computational load by using a truncated Gaussian of size $(\lceil 4\sigma \rceil + 1) \times (\lceil 4\sigma \rceil + 1)$ as the RBF

$$G_\sigma\left(\boldsymbol{x}\right) := \exp\left(\left(-|\boldsymbol{x}|^2\right)/\left(2\sigma^2\right)\right). \tag{2.12}$$

To ensure that the influence of the truncated weighting functions covers all unknown image areas, we adapt the standard deviation σ to the mean free path between known data points [2] according to

$$\sigma = \sqrt{(m \cdot n) / (\pi |K|)}. \tag{2.13}$$

Here m and n denote the image dimensions, and $|K|$ is the number of mask pixels. Our codecs in Section 4.3 benefit from this limitation of the influence area of the known pixels. This allows us to design algorithms with reduced computational complexity and ease of implementation.

Additionally, as done by Achanta et al. [2], we remove the interpolation condition from Eq. (2.10) for our Shepard inpainting. In Chapter 4, we discuss why approximation instead of interpolation benefits our specific application.

After all simplifications, we obtain the final formulation for Shepard inpainting

$$u_i = \frac{\sum_{\boldsymbol{x}_j \in K} G_\sigma\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right) f_j}{\sum_{\boldsymbol{x}_j \in K} G_\sigma\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right)}. \tag{2.14}$$

Since the weighting function only depends on the distance between pixels, it is independent of orientation. Thus, we refer to this strategy as *isotropic Shepard inpainting*. In Section 4.2, we extend this to an *anisotropic* concept with oriented Gaussians which distribute information along dominant image structures [5].

2.2 Data Selection

While the inpainting operator plays a significant role in the final reconstruction quality, choosing the correct pixel data is equally important. The selected data for inpainting-based compression consists of two components: the positions of the pixels and their corresponding values. The optimisation of the pixel positions is called spatial optimisation, whereas optimising their values is called tonal optimisation. A comprehensive overview of different spatial and tonal optimisation methods can be found in Peter et al. [105].

2.2.1 Spatial Optimsiation

Spatial data optimisation or selecting the positions of the mask pixels can be broadly classified into two categories: unconstrained and constrained. Unconstrained mask optimisation approaches optimise only for reconstruction quality without placing additional restrictions on selecting the pixels. On the other hand, constrained optimisation methods also optimise for quality and keep other factors in account, such as coding costs.

Unconstrained Mask Optimisation

Analytic Approaches Belhachmi et al. [19] presented an optimal distribution of known data locations in the continuous setting. The locations in the discrete case can be approximated by dithering on the input image's Laplacian magnitude.

Non-smooth Optimisation Methods Many approaches [22, 31, 55, 94, 95] optimise over a non-smooth function to compute the mask locations for a specific inpainting operator. The locations come with confidence values from the algorithms, which are then binarised and stored.

Stochastic Approaches In stark contrast to the previously mentioned analytic methods, stochastic approaches rely on probability to derive the mask points. The first possibility is probabilistic sparsification [37, 54, 85], which is starting with the whole image and discarding the pixels with the least error iteratively until the desired number of known points is reached. The other possibility is to start with an empty mask and populate it with points where the reconstruction is the largest. The process is done until the desired mask density is achieved. This approach is called probabilistic densification [32, 36, 85].

The reconstruction error from the mask positions derived from all of the approaches mentioned above can be further improved through a post-processing step called *non-local pixel exchange* (NLPE) [85]. NLPE randomly selects a mask pixel and exchanges it with another random non-mask pixel as long as the reconstruction error is reduced.

Semantic approaches Semantic approaches select known data directly based on image features and belong to the oldest inpainting-based compression techniques. They date back to the early approaches of Carlsson [27]. Contemporary semantic codecs are particularly popular for the compression of images with pronounced edges [1, 14] such as cartoons [16, 84, 136, 139], depth maps [49, 58, 74], or flow fields [65]. These methods extract and store image edges and use those as known data for inpainting. Since edges are connected structures, chain codes can be used for efficient encoding.

Although unconstrained mask locations result in very low reconstruction errors, this is not necessarily the best approach from a coding viewpoint. The reason for that is storing these optimised positions can be very expensive. The compression of such unconstrained binary masks has been explored in detail in [91].

Constrained Mask Optimisation

Regular grid approaches These approaches restrict the positions of sparse known data to a fixed grid. Some codecs specify only a global grid size of Cartesian [100] or hexagonal [58] grids, thus steering the density of the inpainting mask. Apart from a few exceptions, such as Peter [100], most regular grid approaches for inpainting-based compression do not offer competitive performance. Additional information, such as edges, can be combined with regular masks [65, 66]. In this work, we consider rectangular grids when dealing with regular masks.

Subdivision approaches Subdivision approaches are the next class of approaches that additionally allow some adaptation to the image, but the points are placed in a specific pattern. Most use error-based refinement of the grid, splitting areas with high inpainting errors into smaller subimages with a finer grid. They store these splitting decisions efficiently as a binary or quad tree. Earlier approaches such as [37, 41, 48, 76] employed a triangular subdivision, while later works [102, 103] use the more efficient rectangular subdivision by Schmaltz et al. [111]. Subdivision allows image adaptivity while offering efficient storage in the form of trees. This motivates us to consider subdivision for our Shepard inpainting-based compression pipelines.

2.2.2 Tonal Optimisation

In addition to optimising the spatial positions of the data, it is also possible to optimise their pixel value. This so-called tonal optimisation [54, 57, 85] modifies the known pixel values to minimise the reconstruction error. Significant reconstruction improvements in the large unknown areas outweigh the errors introduced to the sparse stored data.

Tonal optimisation constitutes a linear least squares problem with a symmetric, positive-definite and dense system matrix for linear inpainting operators such as homogeneous diffusion [85]. Cholesky, LU, and QR factorisations are some direct methods that can be used to solve such systems. In addition, there are also iterative methods, for example, conjugate gradients and LSQR [21]. In the literature, this has been solved, for instance, with gradient descent [54], L-BFGS [31], or by reformulating the problem with the help of so-called inpainting echoes [85]. Hoffman [57] uses Green's functions to express the least squares problem in a way that allows the use of an efficient Cholesky solver. Chizhov et al. [32] deviate from previous approaches by relying on finite elements instead of finite differences. They solve the tonal optimisation problem with an efficient nested conjugate gradient algorithm.

The non-smooth optimisation methods [22, 31, 55, 94, 95] that perform spatial optimisation return a set of mask locations with a non-binary confidence value. Hoeltgen et al. [56] argued that they perform joint spatial and tonal optimisation, as the pixel values can be adjusted through the confidence values.

If the inpainting operator is local [36, 101] or if it can be localised artificially [58, 66, 103], tonal optimisation methods can be tailored to be more efficient. In Section 5.3.2, we take advantage of the fact that Shepard inpainting is local to propose a direct tonal optimisation method.

A simple alternative is proposed by Schmaltz et al. [111]. They visit known pixels in random order and adjust their values to a higher or lower quantisation level in case this yields a lower inpainting error. This method can be used seamlessly with any inpainting operator and quantisation method. The only drawback of this method is the need to compute a full inpainting every time a pixel value is changed, which can be quite time-consuming. Other approaches exist that take quantisation into account. For example, Peter et al. [102] combine inpainting echoes and projecting the computed values to the set of quantised values. In contrast, Marwood et al. [87] implements a stochastic approach to perform tonal optimisation.

Recently, various deep-learning approaches for inpainting and data optimisation have been proposed. Connections between PDEs and neural networks have been explored by Alt et al. [6, 7]. Schrader et al. [112] put forth a fast learning-based spatial optimisation approach for high-resolution images while maintaining similar quality to conventional probabilistic spatial optimisation methods. Learning-based joint spatial and optimisation methods have also been explored by Peter [99, 105]. The rising usage of deep-learning-based approaches in various aspects of inpainting-based compression also makes it sensible for us to explore efficient ways to store and compress the optimised known data.

2.3 Encoding

After selecting the mask locations and corresponding pixel values, we must compress this information to reduce the final file size further. In this section, we review data compression approaches that are not specialised to image data but are still vital to most image codecs. The principle of entropy coders is to convert input data into a more compact representation, which can be converted back to its original form by exploiting the redundancies present in said data. Such general-purpose coders take a sequence of generic symbols from an alphabet as an input. In most cases, this alphabet is some subset of integers.

2.3.1 Huffman Coding

Huffman coding [61] is a classical prefix-free encoding scheme that maps a binary codeword to each alphabet symbol. These codewords are assigned according to a binary tree, which is constructed from the occurrence probabilities of the symbols. As a classical entropy coder, Huffman coding aims to distribute the code lengths to minimise the overall coding cost. The theoretical lower limit for the coding cost of a given symbol distribution $S \rightarrow \{S_1, S_2, ..., S_n\}$ with a probability distribution $p_1, p_2, ..., p_n$ is given by the Shannon entropy [115],

$$H\left(S\right) = \sum_i p_i \cdot log_2\left(p_i\right).$$

Huffman coding is simple, easy to implement, and fast. However, it is inefficient for symbol probabilities that are not close to powers of $\frac{1}{2}$, as all symbols are encoded with an integer number of bits. While Huffman-coding has been superseded by more efficient entropy coders, it is still part of some widely-used image compression codecs such as JPEG [97] and PNG [24].

2.3.2 Arithmetic Coding

In contrast to Huffman coding, arithmetic coding can encode symbols with fractional bit cost per symbol, as it maps the whole source word directly to a binary code word.

In its original form [110], arithmetic coding achieves this by an interval subdivision scheme. Starting with the range $[0, 1)$, each symbol from the alphabet A is associated with a subinterval. The length of these subintervals is proportional to the probability of the associated symbol. Successively, each encoded symbol leads to a refinement of the interval. A dyadic fraction uniquely represents the final interval. Due to the coupling of the interval size to the symbol probabilities, the code length adapts to the symbol distribution.

Since this strategy requires floating-point operations and is therefore slow and prone to rounding errors, we consider the WNC implementation of Witten, Neal and Cleary [135]. It replaces the real-valued intervals with sets of integers. This greatly improves computational performance and allows an easier generation of the binary code word.

Due to its good approximation of the Shannon limit, arithmetic coding is used in many lossless and lossy image compression codecs [23, 26, 38, 48, 60, 64, 87, 111, 120]. It also forms the foundation for more advanced general-purpose encoders, such as context coding and context mixing methods, which we will discuss in the subsequent sections.

2.3.3 Context Coding

Arithmetic coding estimates the symbol probabilities only from the symbol counts, i.e. the number of times that the symbol has been encountered. This pure consideration of the occurrence frequency is referred to as a 0th-order context from which the probabilities are derived. However, more complex contexts can be considered to incorporate more structural information of the input data into the encoding process, thus allowing a better compression performance.

As a direct extension of the 0th-order context, we can use a history of m previously encoded symbols to predict the next symbol. The collection of m previously encoded symbols is called the mth-order context. For images, one can also consider 2D neighbourhoods of varying size (see Fig. 3.4). Using a single context allows us to adapt the probabilities to recurring patterns inside the scope of the corresponding contexts. This enables approaches like Prediction by Partial Matching (PPM) [33], Context Adaptive Binary Arithmetic Coding (CABAC), which is used in HEVC [120] and Extended Block Coding with Optimised Truncation

(EBCOT) which is used in JPEG2000 [26], which have better performance than standard arithmetic coding.

2.3.4 Context Mixing

Context mixing approaches extend upon the idea of context coding. Instead of using only a single context to estimate the symbol probability, the probabilities derived from many contexts can be combined with weighted averaging.

Originating from the pioneering work of Mahoney, the so-called PAQ1 context-mixing algorithm [83], many versions with increasingly sophisticated contexts have been developed. All versions of PAQ encode bits individually rather than the symbols themselves and convert input data into a bit stream accordingly. The small alphabet reduces the number of possible contexts and has the additional advantage that PAQ can be combined with fast binary arithmetic coding algorithms.

As general-purpose compressors, the full versions of PAQ aim to compress mixed data such as combinations of text, audio, executables, HTML code, and many other data types. Thus, many of their additional contexts would not be helpful for our purposes. In the following sections, we consider LPAQ2 by Rhatushnyak [82] since we identified it as the most promising member of the expansive PAQ family.

LPAQ is a lightweight variant of PAQ that uses six local contexts and a prediction context. The local contexts depend on the bits already encoded in the current byte and the previously encoded 4 bytes. The prediction context finds the longest context that was already seen, which is the same as the current one and tries to predict the next bits. A neural network mixes the probabilities which were obtained from the mentioned contexts.

The neural network has a simple structure: an input layer with seven nodes and an output layer that computes the weighted average of the probabilities in the logistic domain. This method of mixing the probabilities is called logistic mixing and uses logarithmic transformations known as stretching and squashing. Their mathematical formulations are given by

$$\begin{aligned} t_i &= \ln\left(p_i / \left(1 - p_i\right)\right), && \text{(stretching)} \\ p &= \sum_i w_i \cdot t_i, && \text{(mixing)} \\ p_{final} &= 1 / \left(1 + e^{-p}\right). && \text{(squashing)} \end{aligned}$$

The algorithm learns the weight for each context during the encoding process by a modified form of gradient descent that tries to minimise the coding cost. This results in larger weights for contexts that give good predictions for the input data. The mixed probability obtained from the neural network is still not perfect. The algorithm refines this probability further using two SSE (Secondary Symbol Estimation) stages. Each SSE stage takes an input probability and then stretches and quantises it. The encoder interpolates the output using a context table to give an adjusted probability. Finally, we use this probability to encode the next bit using arithmetic coding.

However, depending on the local statistics, an individual context can contribute differently in predicting the next bit. Therefore, using the same neural network to encode all bits can be sub-optimal. To this end, LPAQ2 maintains 80 neural networks and chooses which one to use by looking at the match length from the prediction context combined and the bits already encoded in the current byte.

Variants of PAQ have been used successfully in many inpainting-based compression approaches, primarily for storing quantised pixel values [48, 58, 102, 103, 111]. Since the PAQ family of codecs is so successful, we dedicate Section 3.4 to a detailed, separate analysis for them.

2.3.5 Compressing Colour Images

While various inpainting-based codecs can compress colour images [37, 48, 87, 100, 111], the only dedicated colour mode so far is the luma preference (LP) mode for rectangular subdivision (R-EED) [103] with edge-enhancing diffusion (EED) [130]. It relies on the core idea to dedicate a higher budget to the luma channel of YCbCr space than to the colour channels. We adapt this concept to the setting of RJIP. In contrast to our approach, R-EED relies on a more complex inpainting method [132] and a tree-based subdivision scheme to select and store positions of known data. In a broader sense, LP mode resembles the chroma subsampling of JPEG [97].

Vector quantisation is a concept already described by Shannon [115] in his influential early works on information theory. Due to the large amount of research activity in the early 80s and '90s for compression of both visual and audio data, a full review is beyond the scope of this work. We refer the reader to the comprehensive monograph of Gersho and Grey [50] instead.

More recent works that deal with lossy compression and vector quantisation are rare. Venkateswaren and Ramana Rao [126] quantise wavelet coefficients from different sub-bands with vector clustering, while Somasundaram and Rani [118] focus solely on more efficient vector quantisation with a modified k-means clustering. For compression with neural networks, vector quantisation is gaining popularity again [4]. However, it is not applied to colour values but is more generically applied to image features or network parameters to be stored. Zhou et al. [140] combine vector quantisation with inpainting, but in contrast to our work, they quantise blocks of grey value data instead of individual colour values.

Even though they do not deal with colour images, the work of Hoeltgen et al. [73] comes close in spirit to our research. They assess how clustering techniques affect inpainting-based reconstruction from sparse data. However, they only use scalar quantisation. This work confirms, together with the findings of Celebi [29], that the k-means clustering algorithm by Lloyd [77] is one of the best clustering techniques for quantisation. Consequentially, we rely on k-means clustering for our vector mode. We discuss these methods in more detail in Section 5.4.1.

2.3.6 Example End-to-End Compression Methods

In the previous subsections, we have seen the individual components required to build a complete inpainting-based compression method. Now, we discuss some example codecs that use the elements mentioned above.

The first example we discuss is R-EED [103], which works on natural images. They use edge-enhancing diffusion [130, 132] as the inpainting method. The mask points are selected using rectangular subdivision, and the values are optimised using quantisation-aware pixel adjustments [111]. Finally, the mask locations and values are stored using PAQ [81].

Specialised compression methods have been developed for specific kinds of piecewise smooth images. They store some sort of edge information and reconstruct the regions through inpainting. For cartoon-like images, Mainberger et al. [84] store mask points on either side of an edge and perform homogeneous diffusion to reconstruct the full image. The mask locations are stored with JBIG [64], and the values are stored with PAQ. Jost et al. [65] proposed a method to store flow fields, where they store the edges and a uniform mask in the smooth parts and compress them using PAQ. The regions are then reconstructed with homogeneous diffusion. Finally, Jost et al. [66] proposed a general compression method for piecewise-smooth images where the edges are derived by the energy-minimising Mumford-Shah cartoon model [93] and the smooth parts are inpainted with a uniform mask. They found Shepard inpainting as the best option for the inpainting method for their codec.

Another example is the inpainting-based video coder proposed by Andris et al. [10]. Like many other video compression methods, only some frames, known as intra-frames, are stored as images. The rest, known as inter frames, are encoded using flow fields w.r.t. the previous frame and the difference to the actual frame, known as the residual. Inter-frames can only be decoded by decoding the intra-frame and adding it with the corresponding stored flow field and the residual. This codec used homogeneous diffusion inpainting with a rectangular subdivision spatial optimisation. The spatial and tonal data and the flow fields are stored with adaptive arithmetic coding [135], while the residuals are encoded using bzip2 [114]. Andris et al. [9] proposed an extension to this codec to handle full HD images. The intra-frames are reconstructed using homogeneous diffusion with a coarse-to-fine strategy [39] to speed up the inpainting process, while the residuals are inpainted with pseudo-differential inpainting [12]. The spatial optimisation for the intra-frames and flow fields is done with rectangular subdivision. Finally, the entropy coding is performed by Finite State Entropy (FSE) [34], which was used for its speed.

Chapter 3

Sparse Binary Image Compression

3.1 Introduction

Inpainting-based compression methods must store the known pixel locations, also known as the mask and the corresponding pixel values. However, careful and unconstrained optimisation of both the positions and values of the known data [19, 31, 37, 55, 70, 85] is required for maximum reconstruction quality. However, storing optimised mask data is expensive. Depending on the inpainting operator, the distribution of these points aligns with image structures or might lack obvious patterns (see Fig. 3.1). In our case, storing pixel locations is equivalent to storing a sparse binary image. In this chapter, we will propose specialised methods to compress spare binary images and compare them with existing end-to-end image compression methods and general data compression methods. This chapter follows our journal publication, Mohideen et al. [91].

3.1.1 Our Contributions

We aim to offer the first systematic review and in-depth analysis of suitable compression techniques from different fields and propose best practice recommendations for storing unconstrained masks. Our contributions are three-fold:

We start with a systematic review of suitable coding strategies from different fields of compression. We cover dedicated positional coding strategies [38, 87], image compression codecs (JBIG [64], PNG [24], JBIG2 [60], DjVu [23], BPG lossless [26]), and state-of-the-art general purpose encoders (PAQ, LPAQ [81]). In addition, we discuss how these approaches can be applied to different representations of the sparse binary images, such as run-length encoding and methods from data science like Coordinate List (COO) [109] and Compressed Sparse Row (CSR) [109].

Afterwards, we perform an ablation study of context mixing methods. Such approaches combine information from multiple sources of already encoded information to predict the remaining file content, thus increasing compression efficiency. In a top-down approach, we evaluate which parts of complex state-of-the-art methods like LPAQ2 [82] are useful. Moreover, we use a bottom-up strategy to construct a novel context mixing strategy that is simple, yet effective.

Finally, we provide a methodical comparison of all existing and newly proposed methods from the previous two contributions w.r.t. compression efficiency and runtime. In our experiments, we choose the Kodak image dataset [35], which is widely used in compression studies. To obtain meaningful results, we consider different densities of known data (1% through 10%) and different point distributions. We take into account random masks, results from probabilistic sparsification [85] with homogeneous diffusion [62], and densification [3] with Shepard interpolation [116]. In total, this yields a database of 720 diverse real-world test images. This setup allows us to propose best practice recommendations based on our analysis.

Overall, our three contributions allow us to identify the coding methods that yield the best compression performance as well as the best trade-off between speed and coding efficiency.

3.2 Evaluation of Image Compression Codecs

Before discussing specialised methods to compress sparse binary images, we consider well-established image compression methods to provide a performance baseline for the specialised codecs. We discuss and compare five popular lossless image compression methods: PNG, JBIG,

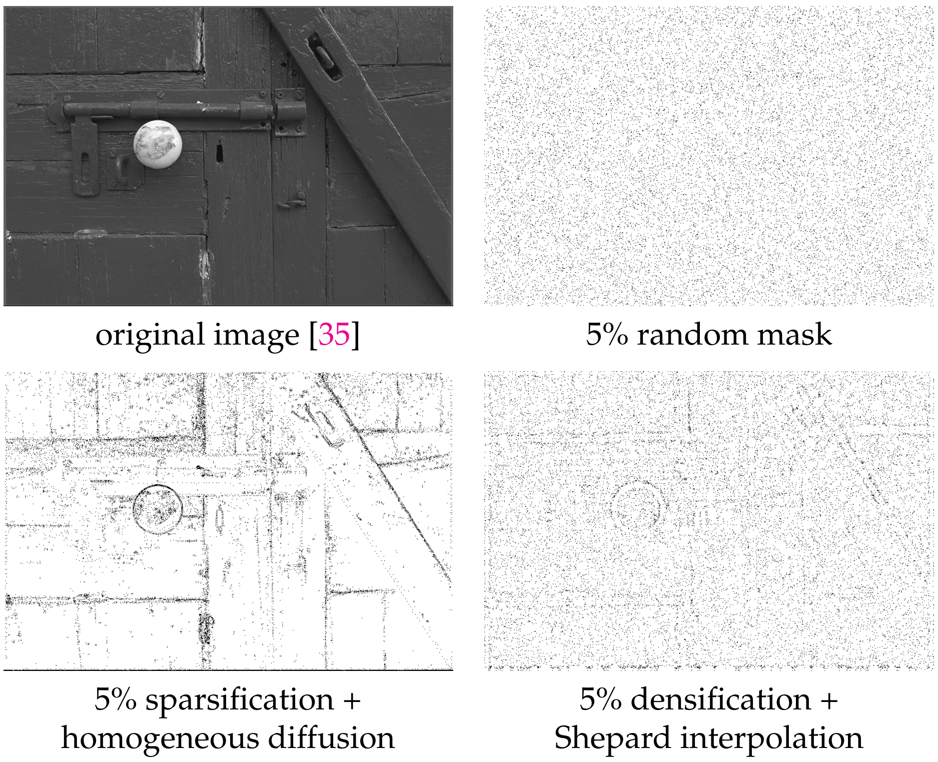

original image [35]
5% random mask
5% sparsification + homogeneous diffusion
5% densification + Shepard interpolation

FIGURE 3.1: We can see here that even though we start with the same image, different point selection strategies or inpainting operators can lead to very different unconstrained mask distributions.

DjVu, JBIG2, and BPG-lossless. Since the salient information of a sparse binary image is the non-zero pixels, the compression ratio as a traditional metric is not the most transparent choice for our evaluation. To this end, we use *bytes per mask pixel* instead, which can be computed as the compressed file size divided by the number of non-zero pixels in the image. Such a normalisation allows us to attribute the cost to the salient image features that need to be stored.

3.2.1 PNG

Portable Network Graphics (PNG) [24] is a widely used lossless image compression codec. PNG compression has two stages: filtering and entropy coding. The filtering stage aims to predict the pixel to be encoded from its already encoded neighbourhood. While the prediction can be wrong, the errors are generally cheaper to compress than the original pixel values due to an overall lower entropy. For predictions, PNG uses only the left, upper and the upper-left diagonal neighbours – either directly as a prediction or by averaging.

The entropy coding uses LZ77 and Huffman coding. LZ77 [141] is a dictionary-based coding method where the subsequent values are encoded by looking for a match in the already encoded values. If a match is found, the difference in position to the current position and the length of the match is encoded. If no match is found, the next value is written explicitly. PNG compresses the values given by LZ77 using Huffman coding.

3.2.2 JBIG

Joint Bi-Level Image Expert Group (JBIG) [64] is a specialised codec from the JPEG family for binary images. JBIG was initially implemented for fax transmissions. It employs progressive coding, which uses a coarse-to-fine strategy to encode the image. On the decoding side, JBIG decodes the coarse scale pixels first to give a low-resolution preview of the image and then the finer scale pixels to display the high-resolution image.

JBIG uses arithmetic coding to encode the lowest resolution pixels, estimating the probabilities using the neighbouring pixels as contexts. It encodes the next finer scale using the neighbouring coarser scale pixels and the neighbouring pixels in the current scale. The algorithm does this until it finishes encoding the finest scale pixels. The cartoon-like image compression codec by Mainberger et al. [84] uses JBIG to store the edge positions.

3.2.3 DjVu

DjVu is an image compression algorithm [23] originally written for compressing images in documents. DjVu includes a bi-tonal image compression algorithm called JB2, which stores the binary image that separates the image into foreground and background. The foreground and background images are then compressed using other methods. If the input is a binary image, DjVu uses only JB2. The algorithm segments the image into connected components of black pixels called marks. The marks are clustered based on similarity. Then DjVu compresses and codes a mark using the previously encoded marks by storing the mark index

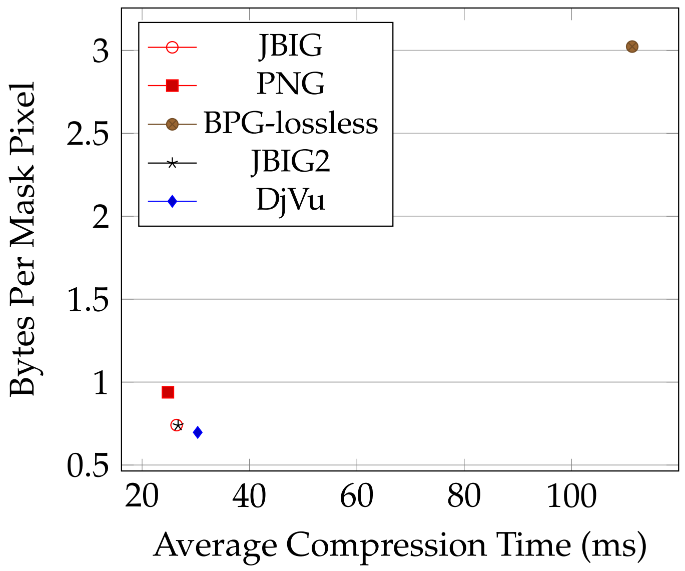


FIGURE 3.2: Performance vs. time comparison of different image-specific compression methods tested on masks derived from the Kodak image dataset. PNG is the fastest, and DjVu is the most efficient. JBIG and JBIG2 give a good trade-off between speed and efficiency.

and relative position. The other marks with no matches are stored using a statistical model and arithmetic coding.

3.2.4 JBIG2

JBIG2 [60] is a binary image codec based on the DjVu JB2 coder. JBIG2 has separate methods for encoding text, which uses pattern matching similar to JB2. It also contains separate methods to encode half-tones and generic data. For our purposes, we need to look at only the generic data coder, which stores two binary images, the coarse image and the refinement bits like in JBIG. The bit-planes are then stored with arithmetic coding.

3.2.5 BPG

Better Portable Graphics (BPG) [26] is a coding method used within the video coding standard HEVC [120]. BPG employs a prediction method that uses three modes: One to model constant regions, one to model directional structures, and one to model smooth areas.

3.2.6 Performance and Runtime Comparisons

In Fig. 3.2, we see that on masks derived from the Kodak dataset, PNG compresses the fastest as it combines simple prediction schemes with Huffman coding. DjVu performs best w.r.t. file size reduction. JBIG and JBIG2 use similar methods to compress non-text data, which explains their comparable performance and compression times. Both give a good trade-off between speed and performance. BPG yields inferior results because our sparse test images violate BPG's assumptions on natural images.

3.3 Representation of Sparse Binary images

As with any other data stream, there are many ways to represent sparse binary images, and how the data is represented impacts the final performance of lossless compression. In the following, we describe and compare common representations originating from different research fields.

3.3.1 Vector Representation

The vector representation is a naive representation that allows the application of many standard sequential entropy coders. We convert the image into a vector by traversing it row-by-row.

3.3.2 Run-length Encoding

One of the most popular methods for the compression of sparse images is run-length encoding. It replaces sequences of identical symbols, so-called runs, by their length.

For our setting, where sparse binary images are considered, run-length encoding becomes even more efficient. We assume that known values are primarily isolated and thus only have to store runs of intermediate zeroes. A zero run can represent consecutive ones.

The image has to be scanned in some order to calculate the run-lengths. Potential options include, for example, column-by-column, row-by-row, diagonal or zigzag, as done in JPEG. We found that the scanning order

1	0	1	0
0	0	0	1
0	1	0	0
0	0	1	0

Vectorised form	1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0
Run-length Encoding (RLE)	0 5 1 2 1
Coordinate List (COO)	(1,1), (1,3), (2,4), (3,2), (4,3)
Compressed Sparse Row (CSR)	1 3 4 2 3 | 2 1 1 1

TABLE 3.1: Example: binary image and its different sparse image representations.

does not significantly impact the final compressed file size. Therefore, we only consider the most straightforward approach, which is column by column.

3.3.3 Coordinate List (COO)

A COO [109] stores each non-zero point as a $(row, column)$ tuple. A differential scheme on the coordinates can also be considered, where we encode the difference in position between the current point and the previous point. However, sorting the points based on their rows and columns, we get an almost identical representation to run-length encoding.

3.3.4 Compressed Sparse Row (CSR)

As another well-known sparse image representation, CSR [109] is mainly used for fast operations on large images in scientific computing. CSR traverses the image row-by-row and stores the column positions of each non-zero element. In addition, the total number of non-zero elements encountered is stored when a complete row is encoded. We slightly modify the basic CSR method to yield better compression results. Instead of storing the total number of non-zero elements encountered, we only store the number of non-zero elements encountered in the previous row. This reduces the range of values needed to be stored, thus reducing the source entropy.

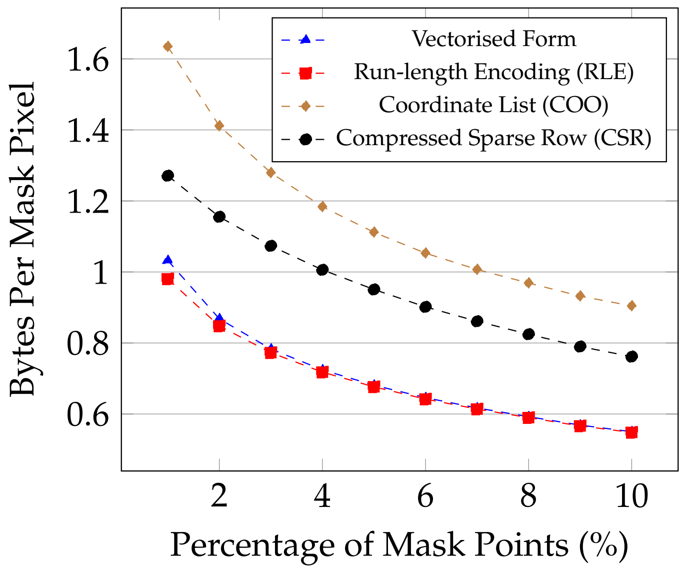


FIGURE 3.3: Compression performance of different representations compressed using LPAQ2 w.r.t. density over the mask images derived from the Kodak image dataset. Vectorised form and run-length encoding are good choices to represent sparse matrices for compression.

3.3.5 Performance Comparisons

In Fig. 3.3, we see that the vector representation and run-length encoding are the better choices as they clearly lead to smaller compressed files than both COO and CSR.

After compression, the file size depends on the initial file size and the file's entropy. The vectorised form has the same probability distribution as the original image. The probability distribution is heavily skewed towards zero for a sparse binary image, yielding a low entropy. This means that the original image is highly compressible, which leads to high performance.

Run-length encoding gives the shortest representation, as seen in Table 3.1, which yields high compression performance, especially for lower densities. On the other hand, COO and CSR have short representations, but not as short as RLE. Also, their entropy is not as low as the vectorised form. These factors combined make COO and CSR non-viable choices for compression.

3.4 An Ablation Study for Context Mixing

In Sections 2.3 and 3.2, we have introduced various coding strategies that are viable candidates for good performance on binary images. However, context mixing is not only the most sophisticated coding strategy, it also has a track record of many applications in inpainting-based compression [48, 58, 102, 103, 111]. Therefore, we investigate which of the many components of these complex methods are helpful for our purpose.

This section describes our setup for a detailed ablation study for context mixing on sparse binary images. In Fig. 3.3, we saw that the vector and RLE representations were best suited for compression. Therefore, we pursue two approaches involving these representations: In a bottom-up strategy, we combine established and new contexts in a novel, minimalistic context mixing codec involving the vector representation. Furthermore, we deconstruct LPAQ2 in a top-down approach to compress run-lengths.

3.4.1 Bottom-Up Context Mixing

The reference point that we start with is the codec used by Marwood et al. [87]. It uses a global context where the probability of the next bit being a one is estimated as V_r/N_r where the remaining number of significant pixels to be encoded is denoted by V_r, and the total remaining pixels to be coded is N_r. From now on, we will refer to this probability as the Marwood context probability. We also considered using the context proposed by Demaret et al. [38], where the context is defined by the number of non-zero pixels in the 12 neighbouring pixels to the current pixel already encoded. However, we found that even though this performs well independently, it does not offer any performance gain when combined with our models.

BPAQ-2D-S

We implemented a lightweight codec BPAQ-2D-S (**S**keletal **PAQ** for **2D** **b**inary data) inspired by the context-mixing structure of Mahoney [81]

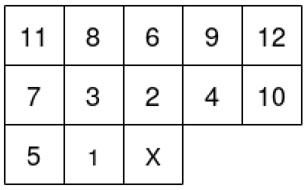


FIGURE 3.4: Local contexts for our version of PAQ. X is the pixel currently being coded. The mth-order context corresponding to pixel X is the pixels that have an index less than or equal to m where $1 \leq m \leq 12$.

where we have only the first order local context (neighbouring pixel to the left) in addition with the Marwood context.

The set of equations is given in Algorithm 1 where n_{10},n_{11},w_1 are the counts for 0 and 1 occurring in the first order context and its weight. The bit currently being encoded is denoted by x. A semi-stationary update for local contexts is done where we halve the count of the non-observed symbol. We update counts this way so that the context probabilities can quickly adapt to local statistics [81]. V_r is initialised to the number of non-zero pixels in the image, and N_r is initialised to the total number of pixels in the image.

When calculating p_{final}, we use a dynamically weighted local probability, a statically weighted local probability, and a statically weighted global probability. The global probability is statically weighted since the counts for 0 and 1 cannot be defined for the Marwood context as it is done for the local context. We noticed through experimental observations that combining static and dynamic weights gives a better performance than just using dynamic weights. We determined the static weights empirically. Finally, we encode the next bit with arithmetic coding using p_{final}.

BPAQ-2D-M

BPAQ-2D-M (**M**ore efficient **PAQ** for **2D** **b**inary data) is an extended version of BPAQ-2D-S. As shown in Fig. 3.4, it has twelve local contexts instead of one. Increasing the order from m to $m+1$ adds one pixel to the context-neighbourhood and allows a better adaptivity to local statistics.

The difference in expressions from BPAQ-2D-S (Algorithm 1) is given in Algorithm 2 where n_{i0}, n_{i1}, w_i are the counts for 0 and 1 occurring in the

Algorithm 1 BPAQ-2D-S

```
1: Compute evidence for the next bit being 0:
       $$S_0 = \varepsilon + w_1 n_{10}$$
2: Compute evidence for the next bit being 1:
       $$S_1 = \varepsilon + w_1 n_{11}$$
3: Compute total evidence:
       $$S = S_0 + S_1$$
4: Compute weighted local probability that the next bit is 1:
       $$p_{dyn} = S_1 / S$$
5: Compute unweighted local probability from the first order context:
       $$p_{stat} = n_{11} / (n_{10} + n_{11})$$
6: Compute Marwood probability:
       $$p_{global} = V_r / N_r$$
7: Update weight for the first order context:
       $$w_1 = \max[0, w_1 + (x - p_1)(S n_{11} - S_1(n_{10} + n_{11})) / S_0 S_1]$$
8: Semi-stationary update of local context weight:
       $$n_{1x} = n_{1x} + 1$$
       $$n_{1(1-x)} = n_{1(1-x)}/2 \quad \text{if } n_{1(1-x)} > 2$$
9: Update counts for the Marwood probability:
       $$V_r = V_r - 1 \quad (\text{if } x = 1)$$
       $$N_r = N_r - 1$$
10: Compute final probability:
       $$p_{final} = 0.4 \cdot p_{dyn} + 0.2 \cdot p_{stat} + 0.4 \cdot p_{global}$$
```

i-th context and its corresponding weight, where $1 \leq i \leq 12$. The static local probability here comes from the fourth-order context instead of the first context in BPAQ-2D-S. Again, it was found out experimentally that the fourth-order context worked best. Apart from these changes, the rest of the expressions stay the same as in BPAQ-2D-S.

Algorithm 2 BPAQ-2D-M

```
1: Compute evidence for the next bit being 0:
       $S_0 = \varepsilon + \sum_i w_i n_{i0}$
2: Compute evidence for the next bit being 1:
       $S_1 = \varepsilon + \sum_i w_i n_{i1}$
3: Compute unweighted local probability from the fourth order context:
       $p_{stat} = n_{41}/(n_{40} + n_{41})$
4: Update context weights:
       $w_i = \max[0, w_i + (x - p_1)(Sn_{i1} - S_1(n_{i0} + n_{i0}))/S_0 S_1]$
```

BPAQ-2D-L

BPAQ-2D-L (**L**ogistic **PAQ** for **2D b**inary data) is similar to BPAQ-2D-M that uses logistic mixing instead of linear mixing where the probabilities are first stretched, and then the final probability is squashed. Unlike BPAQ-2D-M, where the final probability is obtained by statically mixing the mixed local context probability and the Marwood context probability, here the final probability is obtained dynamically where the weights change in each step for all contexts.

In Algorithm 3, $p_1,\ldots,p_{12}$ are the probabilities obtained from the 12 local contexts and p_{13} from the Marwood context. n_{i0}, n_{i1}, w_i are the counts for 0 and 1 occurring in the i-th context and its weight. The bit currently being encoded is denoted by x, and the learning rate α is set to 0.02.

Algorithm 3 BPAQ-2D-L

```
1: Compute context probabilities:
       $p_i = n_{i1}/(n_{i0} + n_{i1})$
2: Stretch context probabilities:
       $t_i = log(p_i/(1 - p_i))$
3: Combine stretched probabilities:
       $p = \sum_i w_i \cdot t_i$
4: Compute the final probability by squashing:
       $p = 1/(1 + e^{-p})$
5: Update context weights:
       $w_i = w_i + \alpha \cdot t_i \cdot (x - p)$
6: Semi-stationary update of local context counts:
       $n_{ix} = n_{ix} + 1$
       $n_{i(1-x)} = n_{i(1-x)}/2 \quad \text{if } n_{i(1-x)} > 2$
7: Update counts for the Marwood probability:
       $V_r = V_r - 1 \quad (\text{if } x = 1)$
       $N_r = N_r - 1$
```

BPAQ-2D-S	Uses 1 local context with linear mixing
BPAQ-2D-M	Uses 12 local contexts with linear mixing
BPAQ-2D-L	BPAQ-2D-M but with logistic mixing
BPAQ-2D-XL	BPAQ-2D-L but with non-contiguous contexts

TABLE 3.2: Summary of the four proposed bottom-up PAQ approaches

BPAQ-2D-XL

Finally, we have BPAQ-2D-XL (e**X**tended **L**ogistic **PAQ** for **2D b**inary data), which is derived from BPAQ-2D-L where the only change is the structure of the local contexts. In the previous models, the higher-order contexts were always contiguously extended from the lower context, as seen in Fig. 3.4. Here, we removed the contiguity conditions and generated contexts that could be disconnected. We only consider the four nearest pixels to the current pixels. Consequently, we would have four different first-order contexts, six different second-order contexts, four different third-order contexts and one fourth-order context. Therefore, we have 15 local contexts in total. The main differences between the proposed bottom-up PAQ models are listed in Table 3.2.

Evaluation: Performance vs. Runtime

In Fig. 3.5, we see that the performance mostly correlates with the complexity of our proposed models. The simplest model, BPAQ-2D-S, is the fastest, and BPAQ-2D-XL is the slowest. Surprisingly, BPAQ-2D-L outperforms all other approaches.

BPAQ-2D-M performs better w.r.t. compression ratio than BPAQ-2D-S due to the additional contexts at the cost of speed. BPAQ-2D-L gains additional compression efficiency, but using logarithmic and exponential functions in logistic mixing slows down the method further. The additional contexts of BPAQ-2D-XL increase its complexity and slightly increase the cost per mask point. Thus, one should either choose BPAQ-2D-L for the lowest bit cost per mask point or the method of Demaret et al. [38] for an excellent mix of compression ratio and speed.

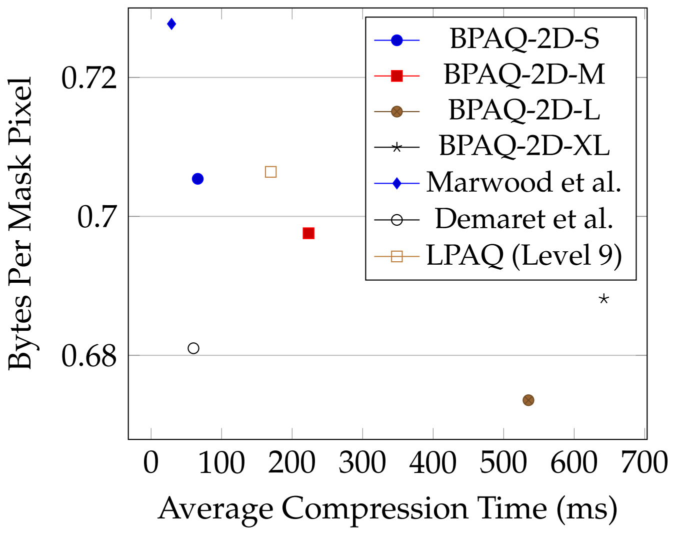


FIGURE 3.5: Performance vs. time comparison of different PAQ-based models that were implemented as part of the ablation study and the codecs proposed by Marwood et al. [87], Demaret et al. [38] and LPAQ2 applied on the image for reference. BPAQ-2D-S is the fastest and BPAQ-2D-L is the most efficient. The method of Demaret et al. gives a good trade-off between speed and efficiency.

3.4.2 Top-down Approaches

LPAQ2 is an excellent entropy coder for a wide range of data and already constitutes a lightweight version of PAQ. However, it is unclear if all of the individual steps of the method contribute to its performance. To identify these parts, we eliminate features step-by-step in a top-down manner to see the performance and runtime behaviour.

We start with the original implementation of LPAQ2 with parameter settings for the highest compression (compression level 9). From there on, we perform multiple reduction steps.

Step 1: We remove the prediction context (no prediction model).

Step 2: We reduce the amount of neural nets from 80 to a single one (single net model).

Step 3: We retain only the intra-byte context and discard the rest (intra-only model).

Step 4: For ULPAQ (ultra lite PAQ), we also discard the second SSE stage (RLE + ULPAQ).

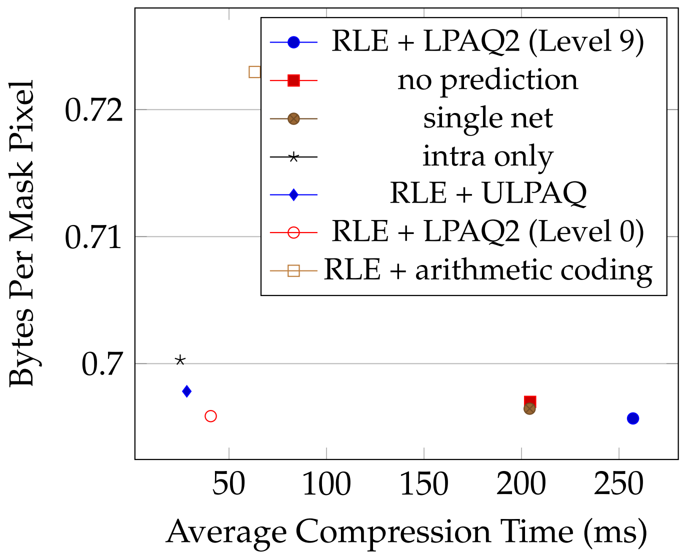


FIGURE 3.6: Performance vs. time comparison of the step-by-step removal of LPAQ2 features to compress run-lengths as part of the ablation study and LPAQ2 (Level 0) (fastest mode) and arithmetic coding for reference. RLE + ULPAQ and RLE + LPAQ2 (Level 0) give a good trade-off between speed and efficiency.

Performance and Runtime Comparisons

A comparison of the aforementioned reduction steps is shown in Fig. 3.6. As the most complex model, the original LPAQ is the slowest of the evaluated methods. The no prediction model and the single network model only have a minor difference in bits per mask pixel, but they are about 20% faster than the original version. This shows that a single neural net without prediction comes at virtually no disadvantage to the more complex full version of the codec.

The intra-only model and ULPAQ are even ten times faster than the original version. Additionally, ULPAQ performs better than the intra-only model. Thereby, the second step of SSE is detrimental to binary input data. We also consider LPAQ level 0, the fastest parameter setting for standard LPAQ. It is not as fast as RLE+ULPAQ but also produces smaller encoded files. In practice, ULPAQ offers a slight advantage in speed, while LPAQ (Level 0) provides a slight advantage in compression ratio.

3.5 Best Practice Recommendations

In this section, we perform a consolidated comparison of the best candidates for sparse binary mask compression from the previous Sections 2.3, 3.3, and 3.4. We consider BPAQ-2D-L and the codec of Demaret et al. [38] from Fig. 3.5, RLE + ULPAQ and RLE + LPAQ2 (Level 0) from Fig. 3.4.2, as well as JBIG2 and DjVu from Fig. 3.2.

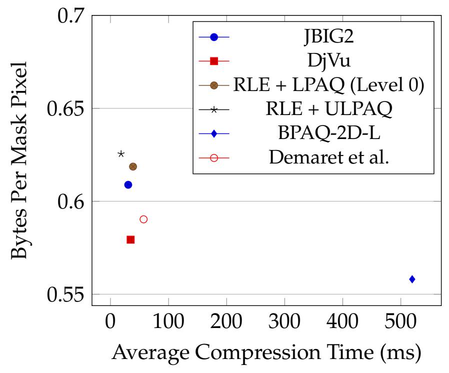
Our evaluation is based on the two essential mask characteristics: Point distribution and density. Both of these aspects are relevant for practical use. The distribution depends on the inpainting operator and mask selection strategy, while the density is closely connected to the desired final compression ratio of the inpainting method. Thus, any best practice recommendations need to consider the behaviour of the algorithms w.r.t. these two factors.

3.5.1 Comparing Compression Performance based on Point Distributions

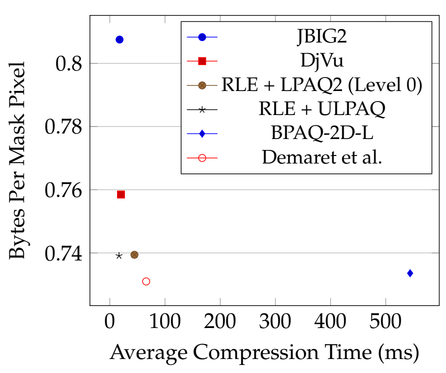
The methods we considered to generate the mask points have varying point distributions; see Fig. 3.1. We want to see if compression performance changes w.r.t. the point distribution. Here, we present the results averaged over all densities from 1% to 10% for different point distributions.

From Fig. 3.7, we observe that the relative rankings of different compression methods change over different types of distributions. DjVu offers fast compression and an excellent compression ratio for homogeneous diffusion masks. If runtime is no concern, BPAQ-2D-L provides the lowest bit cost per mask point.

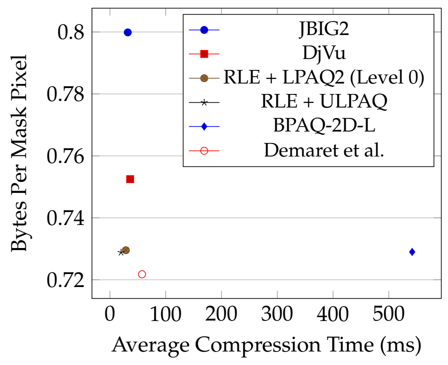
For Shepard and random masks, we see that the two best methods on homogeneous diffusion masks, BPAQ-2D-L and DjVu, perform worse. This is because masks derived from homogeneous diffusion are more structured. BPAQ-2D-L and DjVu can benefit from their ability to detect such patterns. This structure is absent from the masks derived using Shepard interpolation and, in the extreme case, completely random mask distributions. This is also why all the codecs perform better on homogeneous



homogeneous diffusion + sparsification masks



Shepard interpolation + densification masks



random masks

FIGURE 3.7: Performance vs. time comparison of different coders tested on masks derived from different methods but averaged over all densities. For homogeneous diffusion masks, DjVu gives a good performance-speed trade-off where BPAQ-2D-L gives the best performance. For Shepard and random masks, the method of Demaret et al. is the best choice as it is fast and performs well.

diffusion masks than Shepard and random masks. In such a case, the approach of Demaret et al. [38] is a good alternative.

3.5.2 Comparing Compression Performance over Image Density

Next, we evaluate if the relative performance of the selected codecs differs for varying densities. We present the results averaged over all point distributions for masks having densities of 1%, 5%, and 10%. In addition, we also consider an average over all point distributions and all point densities between 1% and 10%.

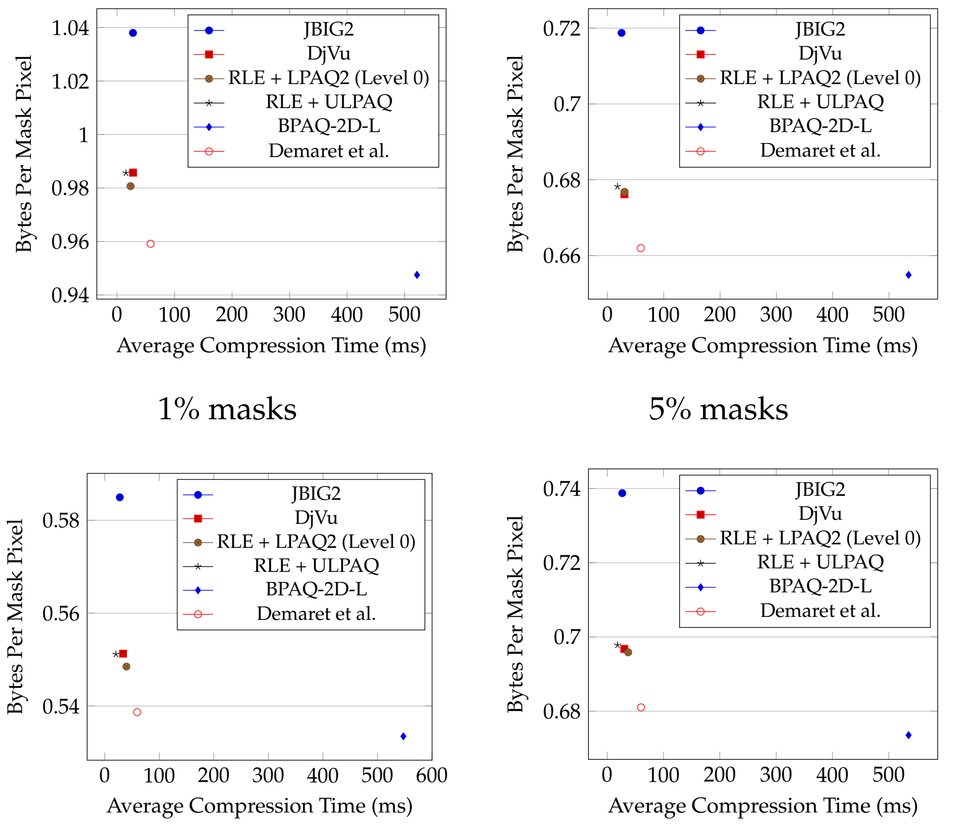
Fig. 3.8 shows that the relative rankings between codecs are consistent over increasing densities and the average of all densities. This implies that the choice of the binary mask compression algorithm has to be only adapted to the distribution but not to the compression ratio.

For time-critical applications, RLE + ULPAQ is the most suitable combination. BPAQ-2D-L, from our ablation study of context mixing, yields the best compression performance at the cost of runtime. Between those two extremes, the method of Demaret et al. provides a good balance between speed and performance.

A general trend can also be inferred from Fig. 3.3: The relative cost of storing positions declines for denser masks. This shows that storing mask positions will decrease the compression performance by a larger amount for sparse masks than dense masks.

3.6 Conclusions

The first systematic study of sparse binary image compression allows us to understand, compare, and improve upon various codecs. With an ablation study, we have determined the aspects of context mixing methods that make them successful for the compression of sparse binary images. As a consequence of the ablation study, we proposed two different classes of methods: one to compress the image itself and the other to compress run-lengths.



1% masks 5% masks

10% masks averaged over 1% to 10% Masks

FIGURE 3.8: Performance vs. time comparison of different coders tested on different densities averaged over masks derived from the Kodak image dataset using sparsification + homogeneous diffusion, densification + Shepard interpolation, and random masks.

Our best practice recommendations provide a foundation for future inpainting-based codecs that use unconstrained masks. Our results from Section 3.5 suggest that optimal mask compression should adapt to the specific point distribution caused by the combination of the inpainting operator and mask selection method. However, the compression method does not need to be changed based on density since that does not impact the relative rankings of the codecs.

Based on our full evaluation, we have concrete recommendations for different use cases: RLE + ULPAQ for time-critical applications, BPAQ-2D-L for highest amount of compression, and the codec of Demaret et al. [38] for a good compromise between compression performance and speed.

These results have exciting implications for future codecs that use unconstrained mask-based inpainting methods. Due to the increased relative cost of storing very sparse, unconstrained masks, we can infer that this is viable only when they offer a massive improvement in inpainting quality over regular or structured masks. It might also be interesting to consider codecs that produce unconstrained masks, which consider not only inpainting quality but also the cost of storing them.

Chapter 4

Compression with Isotropic and Anisotropic Shepard Inpainting

4.1 Introduction

The goal of this chapter is to set a new benchmark in inpainting-based compression that offers a better compromise than existing approaches w.r.t. implementational simplicity, computational efficiency, and approximation quality. We aim at a family of simple and highly efficient codecs that do not require the numerical sophistication of PDE-based codecs while keeping certain quality-critical features such as anisotropy and inpainting mask optimisation. It is based on variants of the classical Shepard interpolation idea [116].

Shepard interpolation provides a fast and easy way to implement sparse image reconstruction by weighted averaging. While the original paper by Shepard [116] proposes inverse distance weighting functions without localisation, we use a variant proposed by Achanta et al. [2], which employs a truncated Gaussian weight function and only approximates the function values in the mask points. We then perform tonal optimisation to maximise the approximation quality.

Due to these deliberate errors in the known data, we choose to perform approximation over interpolation for our inpainting in Section 2.1.4. According to Eq. (2.14), we also inpaint the known data itself, which resembles previous approaches that have addressed this issue through interpolation swapping instead [111]. They explicitly remove disks around the known data in post-processing and inpaint those from the initial reconstruction. This also addresses additional sources for errors in known data: quantisation and noise. Coarse quantisation intentionally reduces the number of permissible pixel values in the grey-level domain, reducing storage costs. On the other hand, noise is unintentional and results in imperfections during image acquisition. Quantisation can be applied as a post-processing after tonal optimisation but might revert some of its improvements. Thus, it is often preferred to account for quantisation already during the tonal optimisation. This chapter follows our manuscript, Mohideen et al. [89].

4.1.1 Our Contribution

Shepard inpainting has not been explored for compression before the conference publication by Peter [100], where it was shown that it allows highly efficient image compression with reasonable quality. Its usefulness has also been confirmed for the compression of piecewise smooth images [66]. In the present work, we extend and improve the results from [100] by fusing the best of both worlds: the high efficiency from Shepard inpainting with two quality improvements from successful PDE-based codecs [48, 111], namely anisotropy [5] and spatial mask adaptation.

We propose a novel anisotropic version of Shepard inpainting, which allows elongated Gaussian kernels to adapt the inpainting direction to the local image structure. Subdivision-based strategies enable us to find better inpainting data than regular masks but are less expensive to store than fully optimised masks. Our experiments show that our codecs can outperform transform-based approaches, particularly at high compression ratios. Moreover, they can offer substantial speed-ups over most implementations of PDE-based inpainting approaches. Our resulting methods are still simple and maintain a favourable trade-off between computational efficiency and reconstruction quality.

4.2 Anisotropic Shepard Inpainting

As seen in Section 2.1.4, Shepard inpainting uses Gaussian functions, which propagate information isotropically in all directions. However, additional directional information is implicitly encoded in the known pixels. This information has been successfully used for anisotropic diffusion inpainting [48, 132]. In the following, we aim to augment our Shepard inpainting with ideas from EED [130] while preserving its simplicity and ease of implementation.

To this end, we introduce *anisotropic Shepard inpainting* [5], which adapts the weighting function to the local directional structure of the available data. We compute gradient information from the known data and use this information to guide the influence function accordingly. Thus, we achieve an anisotropic inpainting effect. In particular, when the known data are arranged regularly, we can compute the gradient information without any overhead.

We propose to modify the weighting function w based on structural information which is encoded in the vector containing the mask pixels $\boldsymbol{f} \in \mathbb{R}^{mn}$ for an image of resolution $m \times n$. To this end, we adapt the weighting function at each mask position $\boldsymbol{x}_j \in K$ to obtain a set of functions w_j. Our anisotropic Shepard inpainting computes the reconstruction u_i as

$$u_i = \frac{\sum_{\boldsymbol{x}_j \in K} w_j(\boldsymbol{x}_j - \boldsymbol{x}_i) f_j}{\sum_{\boldsymbol{x}_j \in K} w_j(\boldsymbol{x}_j - \boldsymbol{x}_i)}. \tag{4.1}$$

The spatially varying weighting functions w_j depend on the local structure of the masked image $\boldsymbol{f}$.

To derive our anisotropic approach, we switch to the continuous setting. We consider a greyscale image $f : \Omega \to \mathbb{R}$, which is fully available on a continuous domain $\Omega \subset \mathbb{R}^2$. The gradient $\boldsymbol{\nabla} f$ encodes the structural information of f, which allows the definition of structure-adaptive weighting functions.

As a weighting function, we choose an oriented Gaussian with standard deviations σ_1, σ_2, and a rotation angle θ. The two standard deviations σ_1, σ_2 determine the major and minor directions of the Gaussian. For $\sigma_1 = \sigma_2$, we want to obtain a rotationally invariant Gaussian corresponding to the isotropic case.

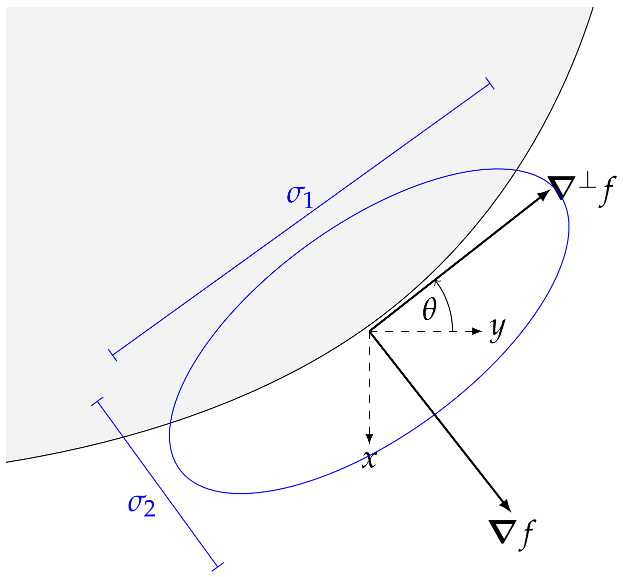


FIGURE 4.1: Local adaptation of the weighting function to the image structure in a continuous setting. The gradient ∇f spans a coordinate system rotated by θ w.r.t. the (x, y)-system. The gradient magnitude shrinks the level lines of the Gaussian kernel (blue) across dominant structures. This gives elliptic level lines with major and minor axes proportional to σ_1 and σ_2, respectively.

A model which fulfils the above properties based on the structural information described by the image gradient $\nabla f = (f_x, f_y)^\top$ in the (x, y)-coordinate system is given by

$$\sigma_1^2 = \sigma^2, \tag{4.2}$$

$$\sigma_2^2 = g\left(|\nabla f|^2\right)\sigma^2, \tag{4.3}$$

$$\theta = \arctan\left(-\frac{f_x}{f_y}\right). \tag{4.4}$$

Here σ is an input parameter determining the base standard deviation of the Gaussian as in the isotropic case. Note that σ_1 and σ_2 are functions of σ and ∇f, but have been abbreviated for readability. In Fig. 4.1, we visualise the relation between these parameters in the continuous setting.

In our experiments, we found the rational Perona–Malik diffusivity [98]

$$g\left(s^2\right) = \frac{1}{1+\frac{s^2}{\lambda^2}}, \tag{4.5}$$

to be a suitable choice for g. It attenuates the variance σ^2 at image locations with dominant structures where the edge detector $|\boldsymbol{\nabla} f|$ exceeds some contrast parameter λ. Choosing the constant diffusivity [63] $g(s^2) = 1$ yields the isotropic Shepard inpainting model.

The angle θ determines the rotation of the deformed Gaussian function. As $\boldsymbol{\nabla} f$ points into the direction of the steepest ascent of f, the deformed Gaussian should be oriented along the orthogonal direction $\boldsymbol{\nabla}^{\perp} f$. In 2D, a vector which is orthogonal to the gradient can be easily constructed, e.g. as $\boldsymbol{\nabla}^{\perp} f = (f_x, -f_y)^{\top}$. The angle of this vector in the respective coordinate system is given by θ in Eq. (4.4).

In the $(\boldsymbol{\nabla}^{\perp} f, \boldsymbol{\nabla} f)$-coordinate system, the resulting Gaussian weighting function should scale the kernel along the principal directions by the standard deviations σ_1, σ_2 given above. Thus, we obtain

$$\boldsymbol{z}^{\top}\boldsymbol{\Sigma}\boldsymbol{z} = \begin{pmatrix} z_1 & z_2 \end{pmatrix} \begin{pmatrix} \frac{1}{2\sigma_1^2} & 0 \\ 0 & \frac{1}{2\sigma_2^2} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \tag{4.6}$$

as an argument for the Gaussian function, where $\boldsymbol{z}$ is the spatial difference between two positions in the $(\boldsymbol{\nabla}^{\perp} f, \boldsymbol{\nabla} f)$-coordinate system. Let us bring this argument to the (x, y)-coordinate system. To this end, we multiply a rotation by θ from the right and its inverse from the left, yielding

$$\boldsymbol{R}_{\theta}^{-1}\boldsymbol{z}^{\top}\boldsymbol{\Sigma}\boldsymbol{z}\boldsymbol{R}_{\theta} = \boldsymbol{d}^{\top}\boldsymbol{R}_{\theta}^{-1}\boldsymbol{\Sigma}\boldsymbol{R}_{\theta}\boldsymbol{d}. \tag{4.7}$$

Here, $\boldsymbol{d}$ is the spatial distance between two positions in the (x, y)-coordinate system. We used the relation $\boldsymbol{z} = \boldsymbol{R}_{\theta}\boldsymbol{d}$ to move the rotation matrices inside the expression.

Expressing the inner matrix with parameters α, β, γ, we obtain

$$\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} = \boldsymbol{R}_{\theta}^{-1}\boldsymbol{\Sigma}\boldsymbol{R}_{\theta}. \tag{4.8}$$

By additionally using the identity $\cos(\theta)\sin(\theta) = \frac{1}{2}\sin(2\theta)$, we obtain

$$\alpha(\theta, \sigma_1, \sigma_2) = \frac{\cos^2(\theta)}{2\sigma_1^2} + \frac{\sin^2(\theta)}{2\sigma_2^2}, \tag{4.9}$$

$$\beta(\theta, \sigma_1, \sigma_2) = -\frac{\sin(2\theta)}{4\sigma_1^2} + \frac{\sin(2\theta)}{4\sigma_2^2}, \tag{4.10}$$

$$\gamma(\theta, \sigma_1, \sigma_2) = \frac{\sin^2(\theta)}{2\sigma_1^2} + \frac{\cos^2(\theta)}{2\sigma_2^2}. \tag{4.11}$$

The resulting Gaussian weighting function takes the distance $\boldsymbol{d} = (d_1, d_2)^\top = \boldsymbol{x} - \boldsymbol{y}$ between the two spatial positions $\boldsymbol{x}$ and $\boldsymbol{y}$. This yields the formula for the anisotropic weighting function,

$$G_{\theta,\sigma_1,\sigma_2}(\boldsymbol{d}) = \exp\left(-\alpha d_x^2 + 2\beta d_x d_y - \gamma d_y^2\right) . \tag{4.12}$$

Here α, β and γ are functions of θ, σ_1 and σ_2.

The discrete implementation of our anisotropic approach is obtained by truncating the Gaussian kernels and using a finite difference approximation for the derivatives d_x and d_y. For regular masks, we can estimate these via the known data. The irregular case is described in Section 4.3.2.

4.3 Compression Pipeline

4.3.1 Regular Grid Codec with Isotropic Shepard inpainting

Our first method aims for maximal simplicity. The *regular grid codec with joint inpainting and prediction* (RJIP) is based on our conference publication [100] with modifications to the tonal optimisation. While it is not image-adaptive, it exploits novel prediction principles with inpainting instead. We store the known data on a regular mask, which means the only storage cost for positional data is the grid size parameter h. This generates minimal overhead. For the grey value data corresponding to the mask positions, we use an equally straightforward uniform scalar quantisation: We map the 8-bit colour values to a reduced range $\{0, \ldots, q-1\}$ by partitioning the tonal domain into q subintervals of equal length.

Shepard inpainting from Eq. (2.14) is implemented by visiting each mask point $\boldsymbol{x}_j \in K$: Its contribution to the numerator is added to the value accumulation map $\boldsymbol{v}$ and the contribution to the denominator is added to the weight accumulation map $\boldsymbol{w}$. Thus, for all points $\boldsymbol{x}_i$ in the truncated Gaussian neighbourhood $\mathcal{N}_j$ of $\boldsymbol{x}_j \in K$, the maps are updated by

$$w_i \leftarrow w_i + G_\sigma(\boldsymbol{x}_i - \boldsymbol{x}_j)\,, \tag{4.13}$$

and

$$v_i \leftarrow v_i + G_\sigma(\boldsymbol{x}_i - \boldsymbol{x}_j) f_j\,, \tag{4.14}$$

with $w_i := w(\boldsymbol{x}_i)$ and $v_i := v(\boldsymbol{x}_i)$. The new inpainting at point $\boldsymbol{x}_i$ can then be computed as

$$u_i = v_i / w_i. \tag{4.15}$$

RJIP employs *joint inpainting and prediction* to decrease the final compressed file size further. Many codecs store the known pixel data directly. However, some compression methods achieve better efficiency by predicting the values to be stored and encoding the prediction error, the so-called *residuals*, instead. Good predictions yield residuals that cluster around zero, thus reducing entropy, which directly translates to reduced storage cost.

We integrate this idea seamlessly into Shepard inpainting. During image compression, the mask points are traversed one by one. If the weight accumulation map w is non-zero at the location of the next mask point to be encoded, an initial prediction can computed through a preliminary inpainting

$$p_i = \hat{v}_i / \hat{w}_i\,. \tag{4.16}$$

Here, $\hat{v}_i$ and $\hat{w}_i$ specify the intermediate maps computed only from previously visited known data. Then we encode the residual between the prediction and the actual mask value as

$$e_i = (p_i - f_i) \bmod q. \tag{4.17}$$

Using a sufficiently large Gaussian as in Eq. (2.13) ensures we predict at least one additional data point in each step. We repeat these steps until all mask points have been visited. Finally, we compress the residuals with a suitable entropy coder. RJIP relies on *finite state entropy* (FSE) [34], a fast alternative to arithmetic coding [135].

For a given compression ratio, RJIP chooses the parameters h and q through a golden-section search to obtain the best reconstruction quality for the desired file size is obtained. While the iterative random walk method that we discussed in Section 2.2.2 is already well-suited for Shepard inpainting due to its locality, it even allows a closed-form solution for the tonal optimisation of individual pixels.

In the following, u_i^{old} and u_i^{new} are the old and new pixel value at $\boldsymbol{x}_i \in K$. We want to find u_i^{new}, such that it minimises the *mean squared error* (MSE). Here, $\mathcal{N}_i$ is the neighbourhood of points around $\boldsymbol{x}_i$, v_j denotes the value accumulation map, and w_j is the weight accumulation map from Eq. (2.14). G_σ is the Gaussian defined in Eq. (2.12), and f_j is the ground truth value at $\boldsymbol{x}_j$. As the tonal error function described at a particular pixel position is convex, an optimal tonal value can be obtained directly by minimising the error function.

This yields the closed-form solution

$$u_i^{\text{new}} = \frac{\sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \frac{G_\sigma(\boldsymbol{x}_j - \boldsymbol{x}_i)}{w_j} \left(f_j - \frac{v_j - G_\sigma(\boldsymbol{x}_j - \boldsymbol{x}_i) u_i^{\text{old}}}{w_j} \right)}{\sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \frac{G_\sigma(\boldsymbol{x}_j - \boldsymbol{x}_i)^2}{w_j^2}} . \tag{4.18}$$

A detailed derivation for this improved tonal optimisation is presented in Section 5.3.2. To account for quantisation, we project these optimal values to the set of admissible quantised values. We iterate over all mask points multiple times until the process converges. As we do not have to compute the inpainting for the entire image every time we change a pixel value, the tonal optimisation for isotropic Shepard inpainting is highly efficient. Our experiments in Section 4.4.2 demonstrate that it is competitive to sophisticated diffusion-based tonal optimisation despite its simplicity.

4.3.2 Subdivision Codec with Anisotropic Shepard inpainting

Subdivision masks offer a good balance between image adaptivity, coding costs, and complexity. We adopt this concept from Schmaltz et al. [111]

by starting with the whole image as a single block and placing a mask point in each corner of the image. If the reconstruction error in a block exceeds a threshold parameter, we split the block in half along its largest dimension and add mask points at each corner of the smaller blocks. This process is repeated until no block violates the error threshold. We store the splitting decisions efficiently in the form of a binary tree.

To overcome the issue of approximating derivatives on a non-uniform grid, we first perform an isotropic Shepard inpainting on the mask and then compute derivatives on the inpainted image. With these derivatives, we compute the final anisotropic inpainting.

Uniform mask codecs use a global Gaussian standard deviation σ in Eq. (2.13). However, a global σ that ensures no holes in sparse image regions leads to overly smooth reconstructions, especially in regions with high mask density. Therefore, in the case of subdivision mask codecs, we adapt σ to the local density of the mask. Unfortunately, explicitly storing individual variance values for each mask point would be too costly. Instead, we derive the local variance from the mask itself following [127]:

$$\sigma_k = (\log(1 + A_k))^p \, . \tag{4.19}$$

Here, σ_k is the variance at mask point k, and p is a constant. We obtain A_k from a Voronoi decomposition [13]. It partitions the image into Voronoi cells: For each mask point $\boldsymbol{x}_k$, the corresponding cell contains all image points closer to $\boldsymbol{x}_k$ than any other mask point. Therefore, the area A_k of a Voronoi cell provides a way to estimate the local mask density and, thus, the desired standard deviation. Instead of optimising and storing individual values for σ, we optimise and store p instead.

We also have to optimise for the diffusivity function's contrast parameter λ. As λ only determines the degree of anisotropy for all kernels, it does not need to be adapted locally.

Unlike RJIP, which had a target ratio as a model parameter, we optimise our subdivision and quantisation w.r.t. a target splitting error. We first find the quantisation parameter q. To that end, we consider the curve that maps quantisation levels to the corresponding quantisation error for the original image. This curve is decreasing because an increasing number of quantisation levels implies a reduction of the quantisation

Algorithm 4 Summary of the subdivision codec with anisotropic Shepard inpainting

```
1: Compute $q$ from the quantisation error curve.
2: while number of nodes split $> 0$ do
3:     Place mask points at leaf nodes.
4:     while $n <$ iter_max, iter_max $\in \mathbb{N}$ do
5:         Optimise for $\lambda$ and $p$.
6:         Perform tonal optimisation.
7:     Compute reconstruction error at each leaf node.
8:     Split leaf nodes with error $>$ target splitting error.
9: Compress tree and mask values with LPAQ.
```

error. We select the value of q such that the curve has a derivative value of 1. This value of q is the point of diminishing return, after which increasing q does not result in a significant decrease in quantisation error.

After fixing q, we perform the actual subdivision. We start at the root of the tree and reconstruct the image with just the four corner points of the image with anisotropic Shepard inpainting with optimised p and λ and then tonal optimisation. The parameter optimisation and tonal optimisation are alternated to adapt the parameters and the tonal values to each other so that the inpainting quality is increased. If the reconstruction error is higher than the target splitting error, we split the node and go to the next level. This process of adjusting the parameters p and λ, the tonal optimisation, and node splitting is repeated for each tree level by going deeper into the tree until all sub-images or leaf nodes of the tree have a reconstruction error lesser than the target splitting error. Finally, the subdivision tree and the mask values are compressed by applying LPAQ2 [82]. An overview of the algorithm can be seen in Algorithm 4.

4.4 Experiments

In this section's four parts, we systematically evaluate our inpainting operators and compression pipelines. In the first part, we compare our proposed and existing inpainting operators on a synthetic disc image. Then we compare their runtime scaling behaviour w.r.t. the number of pixels on a single image of the Sintel [108] database. In the third part, we run our uniform mask-based codecs on the *trui* image and the greyscale version of the Berkeley dataset [86] and compare them to JPEG and JPEG2000. The final part is dedicated to our subdivision-based codecs, where we consider performance on the greyscale version of the Kodak dataset [35] with JPEG as a reference.

4.4.1 Comparing Inpainting Operators

In Fig. 4.2, we consider the inpainting quality and computational effort on a synthetic binary disk image with a uniform mask of density 11.11%. In addition to our isotropic Shepard and anisotropic Shepard inpainting, we also consider homogeneous diffusion inpainting [63] as a baseline since it is widely used in compression applications due to its relative simplicity compared to more sophisticated diffusion inpainting. Thus, it is the direct competitor for our Shepard inpainting. To ensure a fair comparison w.r.t. timing, we use several contemporary solvers for homogeneous diffusion [32, 57].

From Fig. 4.2, we can see that the isotropic Shepard inpainting result yields slightly blurrier edges than homogeneous diffusion. However, the inpainting is faster by one to three orders of magnitude. Tonal optimisation can even be faster by up to five orders of magnitude compared to the approach of Hoffmann [57]. Even the recent highly efficient finite elements method is still slower by one order of magnitude. This is a direct effect of the closed-form solution from Section 4.3.1 for isotropic Shepard inpainting. Thus, it constitutes a good alternative to homogeneous diffusion inpainting for time-critical applications while being significantly easier to implement.

The anisotropic version of Shepard inpainting yields much sharper results than its competitors. However, this comes at the price of a higher computational load. For pure inpainting, anisotropic Shepard is still faster than homogeneous diffusion by one to three orders of magnitude depending on the solver. However, homogeneous diffusion can be faster for tonal optimisation. Here, the anisotropy prevents the efficient closed-form solution and requires a fall-back to a simple trial-and-error algorithm. Thus, anisotropic Shepard is a good choice for pure inpainting applications and compression which requires high quality.

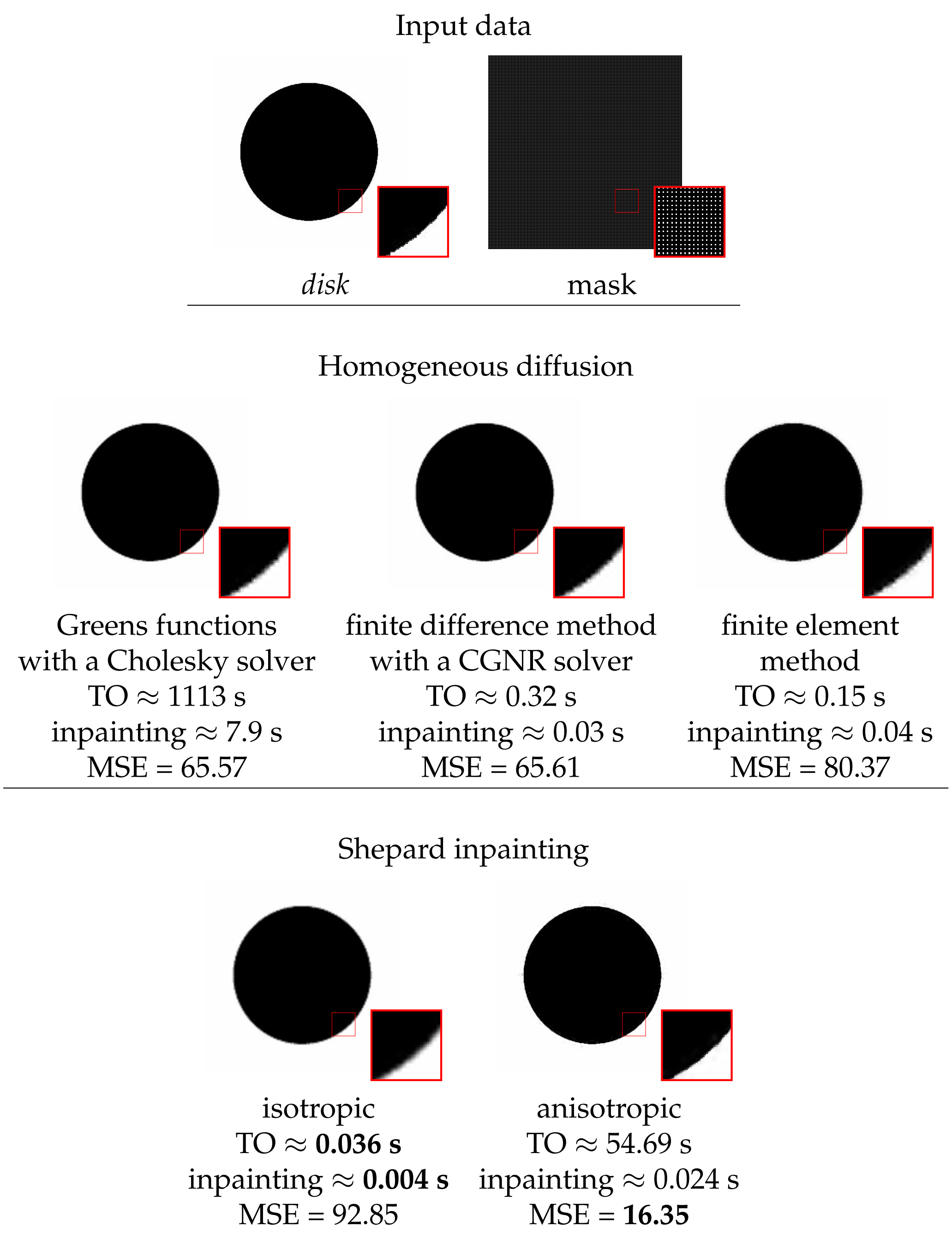


FIGURE 4.2: We compare the inpainting quality, time for tonal optimisation (TO), and inpainting time of Shepard inpainting to several contemporary solvers for homogeneous diffusion inpainting on a synthetic image of a disk of size 400×400. All approaches use the same uniform mask of density 11.11% (1 in 9 pixels). The experiments were run on a single core of a single core of an Intel Core i7-6700A@3.40GHz with 32 GB RAM. The experiment highlights sharper results of anisotropic Shepard inpainting compared to all competitors at an inpainting speed comparable to homogeneous diffusion. Localised isotropic Shepard inpainting can be up to an order of magnitude faster than non-localised homogeneous diffusion inpainting at a similar quality.

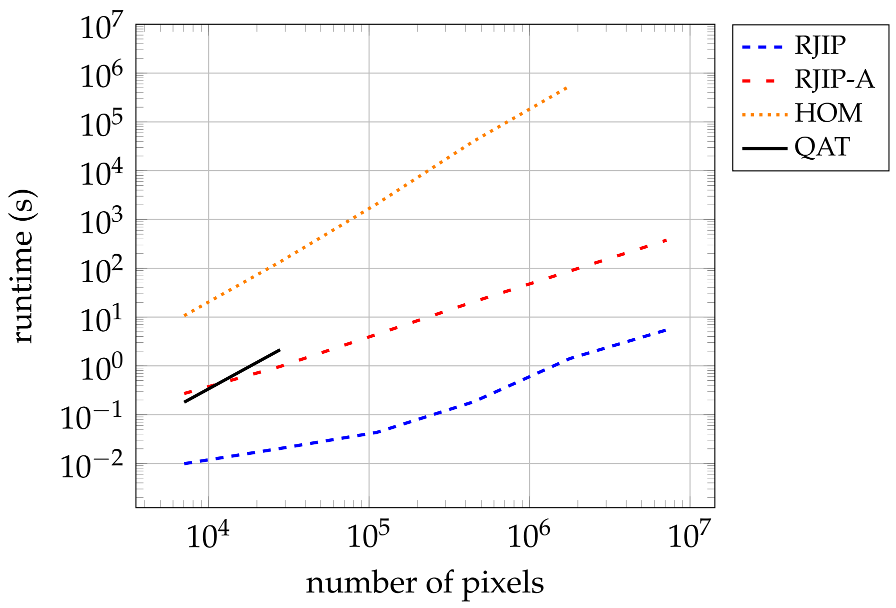


FIGURE 4.3: We compare the runtime scaling behaviour of our pipelines with homogeneous diffusion on five downsampled versions of frame 917 of the 4K CinemaScope movie *Sintel* (4096 × 1744). Here, both axes are presented in a log scale. The experiments are conducted on a single core of an Intel Core i7-6700A@3.40GHz with 32 GB RAM.

4.4.2 Timing Experiments

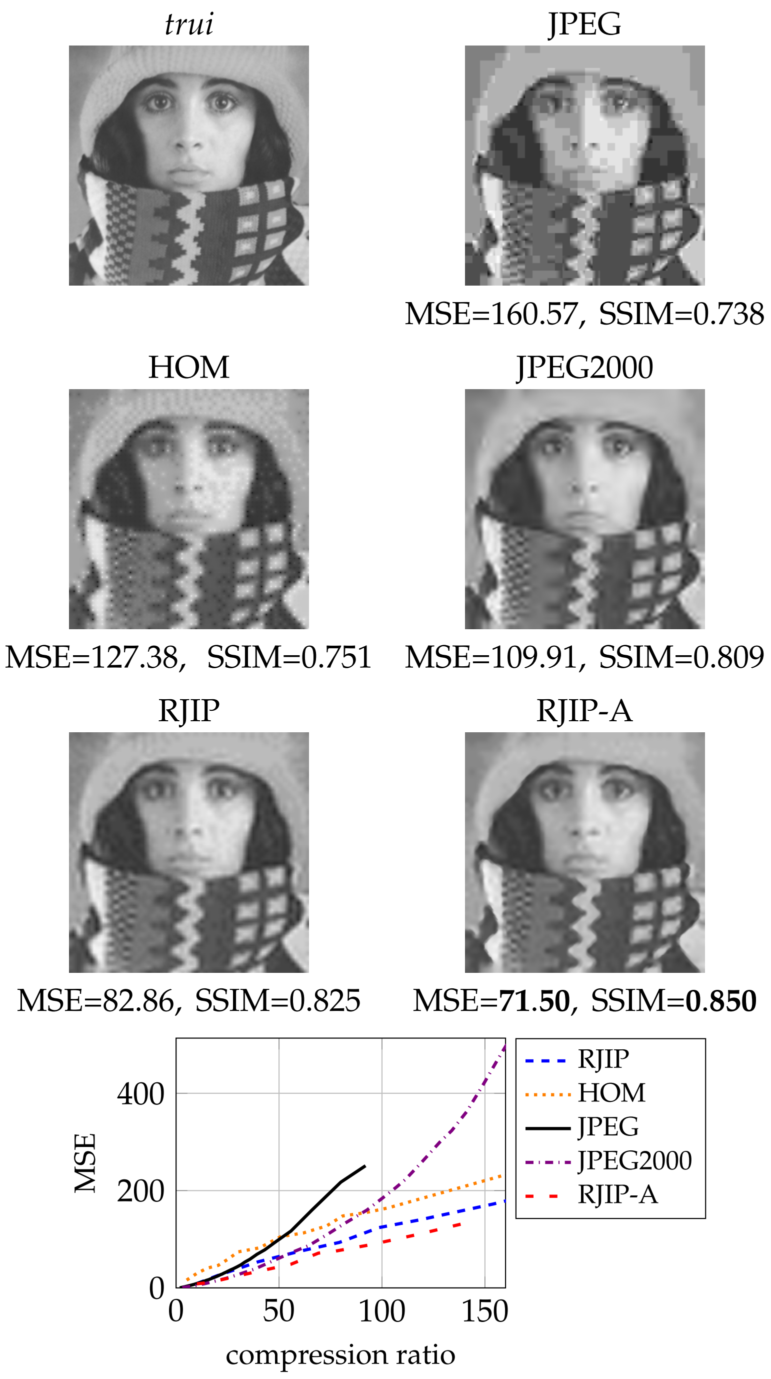
In the following, we investigate the scaling behaviour of our Shepard inpainting-based compression methods and homogeneous diffusion with the number of pixels. We consider RJIP and a uniform mask codec with anisotropic Shepard inpainting.

Our experiments consider a single compression with a fixed sampling distance $h = 4$ ($\approx$ 4% mask density) and $q = 32$ quantisation levels. We implemented a version of our RJIP codec, which uses homogeneous diffusion solved through a conjugate gradient scheme. We refer to this homogeneous diffusion version of the codec HOM. It is to be noted that our proposed tonal optimisation does not apply to HOM, which, therefore, uses the simple iterative pixel adjustment scheme. From Fig. 4.3, we can see that RJIP is faster than HOM by up to 5 orders of magnitude. The compression times for RJIP range from 0.009s for a 128 × 55 image to 5.45s for a 4K image.

In contrast to RJIP, it is unpopular to localise HOM by choosing a finite stopping time, since one is usually not willing to accept the quality deterioration. Thus, the tonal optimisation is more time-consuming for non-localised homogeneous diffusion than for the localised RJIP. Peter et al. [102] proposed an alternative *quantisation-aware tonal optimisation* (QAT) which increases the speed at the cost of high memory consumption. We could not test QAT for images larger than 512×218, as our test machine ran out of memory. RJIP is still 2 to 3 orders of magnitude faster than this specialised algorithm.

We also implemented a uniform mask compression method like RJIP but improved it with anisotropic Shepard inpainting, which we call RJIP-A. In addition to the parameter optimisation for the grid size and the quantisation parameter, we also run an optimisation to find the contrast parameter λ and the variance of the Gaussians σ similar to our anisotropic Shepard codec on trees. We fall back to the iterative random walk method for tonal optimisation, as a clean, optimal solution for the tonal value cannot be computed here as in the case of RJIP. We also alternate the tonal and parameter optimisation to further increase quality as in Section 4.3.2. Finally, as we need all neighbouring pixels to compute derivatives and consequently the inpainting at that pixel, we cannot perform joint inpainting and prediction as used in RJIP. Therefore, we compress the pixel values with LPAQ.

We can also observe that RJIP-A is about two orders of magnitude slower than RJIP. This can be easily explained by the fact that, in addition to searching for h and q, we have to search two additional parameters (λ, σ) for every combination of (h, q). Each inpainting is slower due to the computations required to calculate the Gaussian kernel at each point. Additionally, as changing a pixel requires us to update the neighbourhood derivative information, tonal optimisation is slower than RJIP. Despite all the additional computations, we are still faster than HOM by about one to two orders of magnitude. Furthermore, we can see that our anisotropic codec is slightly slower than QAT for very low resolutions but quickly becomes more efficient as resolution increases. Consequently, the additional computations required for anisotropic Shepard inpainting results in a considerable increase in reconstruction quality.



Comparisons on the *trui* image

FIGURE 4.4: We compress *trui* with different compression methods at a ratio of 70:1 and compare them with the MSE error measure. Additionally, we display the SSIM [128] scores for the presented images. We can see that our Shepard inpainting-based methods do not present any unpleasant artefacts. From the rate-distortion curves, we can observe that our anisotropic Shepard codec with uniform masks outperforms JPEG2000 over most compression ratios, while the base RJIP codec outperforms JPEG2000 at a compression ratio of 60.

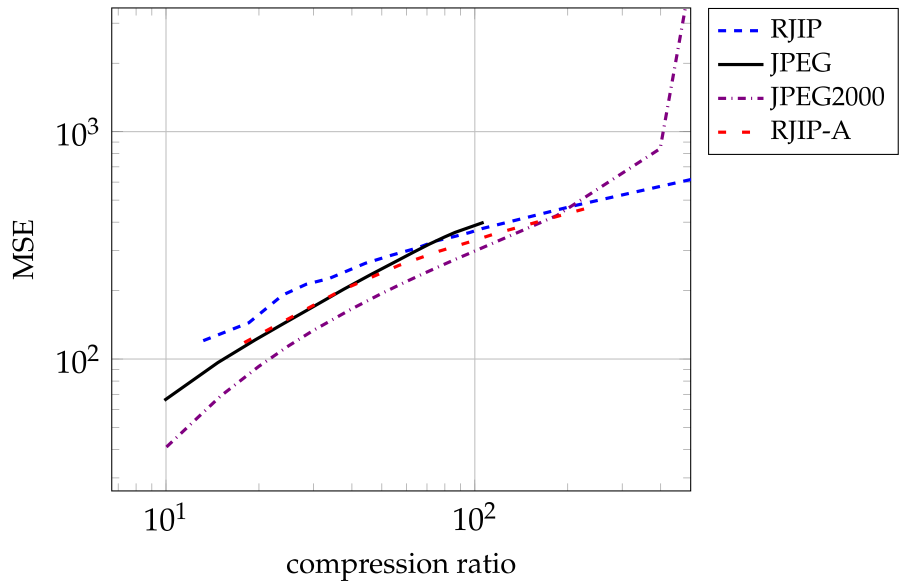


FIGURE 4.5: On the Berkeley database with textured images, RJIP-A is competitive to JPEG at low compression ratios, whereas RJIP and RJIP-A outperform JPEG and JPEG2000 at high compression ratios. Both axes in the plot are presented in a log scale.

4.4.3 Comparing Uniform Mask Codecs

In this set of comparisons, we compare the base RJIP codec with isotropic Shepard inpainting on uniform masks to RJIP-A, the uniform mask codec with anisotropic Shepard inpainting. This allows us to evaluate the relative performance of the anisotropic Shepard operator on the piecewise smooth test image *trui* image and the 500 textured images of the Berkeley dataset [86]. Moreover, we also compare against JPEG and JPEG2000.

From Fig. 4.4, we observe that on the piecewise smooth *trui* image, RJIP-A performs the best across almost all compression ratios, even compared to JPEG2000. It benefits from its superior reconstruction of directional structures with anisotropic Shepard inpainting.

For the Berkeley dataset, this advantage is also visible for low to medium compression ratios in Fig. 4.5: RJIP-A outperforms its isotropic counterpart RJIP and is competitive with JPEG. At higher compression ratios, both Shepard codecs yield very similar quality and are able to beat both JPEG and JPEG2000. These very sparse mask grids contain less reliable information on anisotropy, which can cause the anisotropic Gaussian kernels to degenerate to an isotropic setting.

The more favourable rate-distortion behaviour at higher compression ratios compared to transform-based compression results from the fact that the number of mask points to be stored does not need to be reduced proportionally to the compression ratio. At high ratios, coarse quantisation can be used to reduce the coding cost, and the smooth inpainting is able to restore a wider range of grey values than that of the mask pixels. This makes our codecs particularly suited for compressing images at high compression ratios.

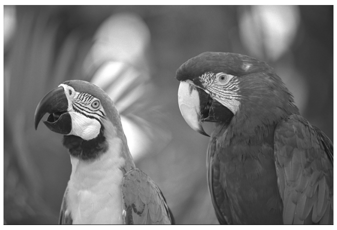
4.4.4 Comparing Subdivision-Based Codecs

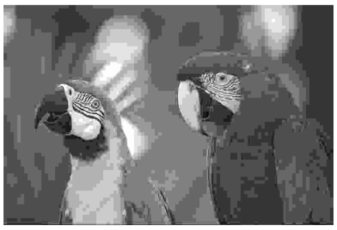
In our final comparison, we consider the rate-distortion behaviour of our subdivision codecs for isotropic and anisotropic Shepard inpainting on the greyscale Kodak dataset. We also include the uniform mask codecs from the previous set of experiments and consider JPEG as a reference.

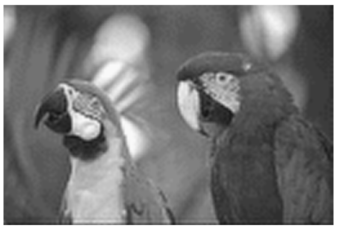
Our uniform mask codecs use multiple target compression ratios while we specify different target splitting errors for our subdivision codecs to steer the rate-distortion trade-off. For this set of experiments, we present images that show representative performance for different image content and different behaviour of our algorithms in Fig. 4.6. The rate-distortion curves for all images in the Kodak dataset can be found in the supplementary material.

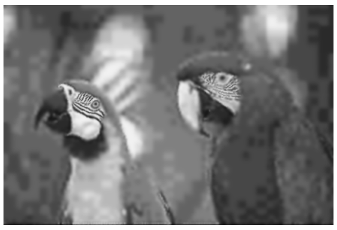
Fig. 4.6 illustrates that for images with a dominant foreground and a homogeneous background, the subdivision codecs outperform their regular grid counterparts. In such images, the subdivision codecs benefit from denser known data in textured regions. This also leads to a higher accuracy of the derivative approximations for anisotropic Shepard inpainting in such dense regions, yielding a better overall reconstruction quality. Even at higher compression ratios, our subdivision codecs can retain more structures compared to the original RJIP codec.

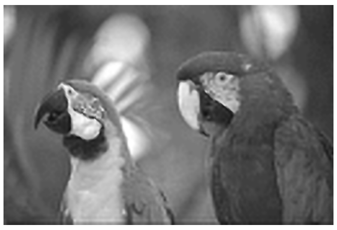
On the other hand, for images that are highly textured overall, the advantages of the subdivision codec diminish. Since all regions are similarly detailed, it produces uniformly distributed masks and thus degenerates

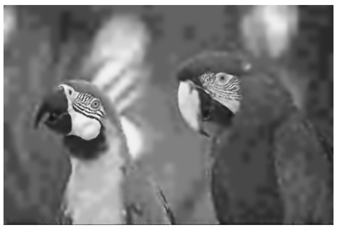

kodim23

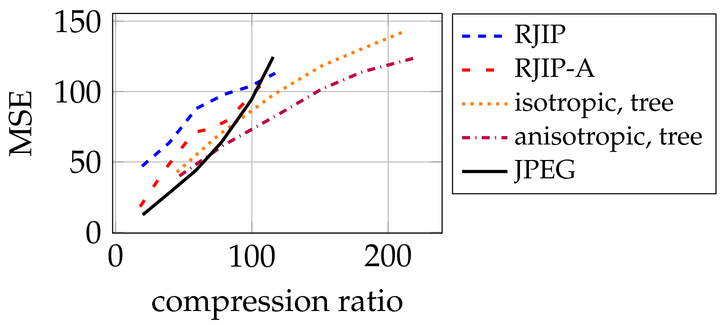

JPEG
ratio 116:1, MSE = 124.63


uniform isotropic (RJIP)
ratio 117:1, MSE = 113.06


tree isotropic
ratio 116:1, MSE = 97.82

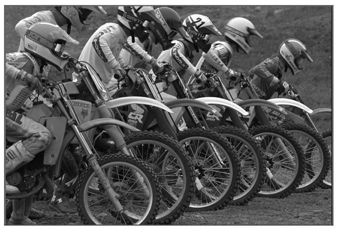

uniform anisotropic (RJIP-A)
ratio 109:1, MSE = 107.10

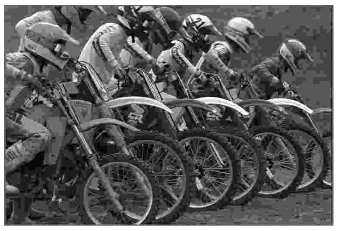

tree anisotropic
ratio 120:1, MSE = **83.93**

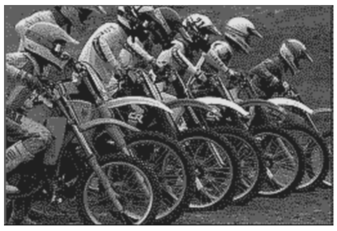


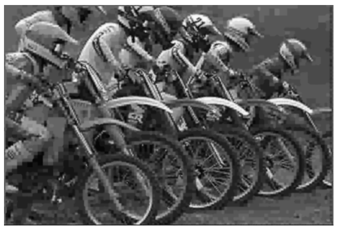
rate-distortion comparisons for *kodim23*

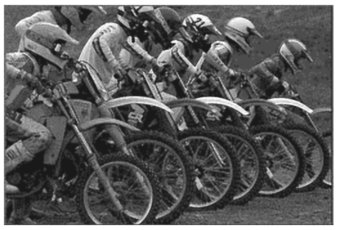
FIGURE 4.6

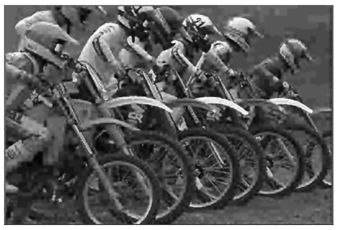

kodim05

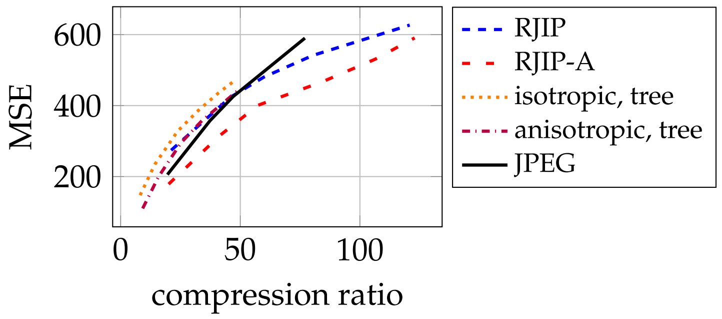

JPEG
ratio 37:1, MSE = 357.34


uniform isotropic (RJIP)
ratio 46:1, MSE = 423.26


tree isotropic
ratio 41:1, MSE = 433.64


uniform anisotropic (RJIP-A)
ratio 41.56:1, MSE = **316.24**


tree anisotropic
ratio 42:1, MSE = 401.38



rate-distortion comparisons for *kodim05*

FIGURE 4.6: Our anisotropic Shepard inpainting-based codecs perform better than their isotropic counterparts overall and outperform JPEG at higher compression ratios. The plots show that the subdivision-based codecs perform better than the regular mask-based codecs if the image has a clear foreground subject since they can adapt the mask such that the detailed regions are reconstructed better. On the other hand, the regular mask-based codecs perform better if the image is textured overall.

to the uniform codec. However, it still requires significant overhead for the tree structure and thus performs worse than its simpler counterpart. This suggests switching between both coding archetypes depending on the image content.

4.5 Conclusions

We have presented a family of simple image compression methods based on Shepard inpainting. This family of codecs relies on new ideas, such as a more powerful anisotropic extension of Shepard inpainting and the novel concept of joint inpainting and prediction. Our codecs are easy to implement and can be orders of magnitude faster than classical PDE-based inpainting methods due to the locality of Shepard inpainting. In particular, codecs with isotropic Shepard inpainting offer efficiency and ease of implementation, while our anisotropic Shepard operators balance efficiency and reconstruction quality well.

Our anisotropic Shepard inpainting offers better quality than isotropic Shepard inpainting, especially at lower compression ratios. For piecewise smooth images, our family of codecs is competitive with JPEG and JPEG2000. On textured data it can yield competitive results to transform-based compression at high compression ratios.

We are working on extending RJIP and our anisotropic and subdivision codecs to colour images by employing dedicated strategies that exploit human perception. As Shepard inpainting excels on piecewise smooth data, we hope it contributes to the compression of depth maps or flow fields. First attempts that followed our earlier conference publication [100] already yielded promising results [66].

Chapter 5

Compression of Colour Images

5.1 Introduction

In the previous section, we discussed image compression with Shepard inpainting. However, the codecs were designed for single-channel grayscale images. Popular transform-based codecs have dedicated colour modes to compress multi-channel images. Although each codec [97, 121] implements it differently, the underlying idea for dedicated colour modes is that structural information is visually more important than colour. By discarding more information in the colour channels, they can store the structure information more accurately. Inpainting-based codecs usually perform scalar quantisation on each channel, which is inpainted separately. An exciting side-effect of inpainting is that it tends to fill in gaps not only in the spatial domain but also in the colour domain. Till now, colour modes for inpainting-based codecs have not been explored much. In addition, inpainting-based compression methods that create sparse colour palettes through vector quantisation have not yet been proposed. This chapter follows our manuscript, Mohideen et al. [90].

5.1.1 Our Contribution

We aim to evaluate different concepts for inpainting-based colour image compression. To this end, we propose two new colour modes for the recent regular grid coding with joint inpainting and prediction (RJIP) codec [100]. Our first mode adapts a state-of-the-art colourisation-based concept [103] in YCbCr mode to the RJIP setting: We dedicate a higher budget to the image structure than to colour and augment it with efficient post-processing specifically tailored to fast Shepard interpolation [116]. Additionally, we propose the first full vector quantisation mode for inpainting-based compression. Our evaluation on the Kodak database [35] reveals that both colour modes significantly outperform the original RGB mode.

5.2 Review: Inpainting-based Compression with Scalar Quantisation

5.2.1 Inpainting

At their core, all inpainting-based image compression codecs rely on interpolation from sparse image data. For a typical inpainting problem in RGB space, the colour image $\boldsymbol{f} : \Omega \to \mathbb{R}^3$ is only known at a few locations, the inpainting mask $K \subset \Omega$. The missing parts of the image domain Ω need to be reconstructed during decoding.

For the isotropic Shepard inpainting in RJIP [100], computing an unknown pixel value $u_c(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \Omega \setminus K$ for each channel $c \in \{R, G, B\}$ comes down to a simple weighted averaging of the known data according to

$$u_c(\boldsymbol{x}_i) = \frac{\sum_{\boldsymbol{x}_j \in K} w(\boldsymbol{x}_j - \boldsymbol{x}_i) f_c(\boldsymbol{x}_j)}{\sum_{\boldsymbol{x}_j \in K} w(\boldsymbol{x}_j - \boldsymbol{x}_i)}, \tag{5.1}$$

as discussed in Section 2.1.4. Following Achanta et al. [2], we use a truncated Gaussian w with standard deviation $\sigma = \sqrt{(m \cdot n)/(\pi |K|)}$ for a discrete $m \times n$ image where $|K|$ is the number of mask pixels.

5.2.2 Data Selection and Storage

Each inpainting-based codec requires an adequate strategy to select and encode the inpainting mask. RJIP stores the known data on a regular grid with grid size parameter h. This approach is fast and generates little overhead but results in many colour values that need to be stored. RJIP compensates for this fact by *joined inpainting and prediction*: While decompressing the image, it performs a partial inpainting whenever a new mask point is decompressed and uses this to predict the remaining mask points. Thus, only prediction errors need to be stored.

For the colour data corresponding to the mask positions, existing inpainting-based codecs use *uniform scalar quantisation* in each channel: They map the 8-bit colour values to a reduced range $\{0, \dots, q-1\}$ by partitioning the tonal (i.e. colour value) domain into q subintervals of equal length, limiting the number of different colours for the known data to q^3. These quantised values are then stored with a suitable entropy encoder. RJIP relies on finite state encoding (FSE) [34], a fast coder similar to arithmetic coding.

For a given compression ratio, RJIP chooses the parameters h and q to obtain the best reconstruction quality for the desired file size.

5.2.3 Tonal Optimisation

RJIP benefits from *tonal optimisation* in post-processing. It performs iterative random walks over all $3|K|$ R, G, or B values, adjusting them to a higher or lower quantisation level if this yields a lower inpainting error. Even though this introduces a bias to the sparse stored data, it can significantly increase the overall reconstruction quality in the large unknown areas. We need to adapt this step for vector quantisation.

5.3 RJIP with Luma Preference Mode

5.3.1 File Size Budget Distribution

As a first colour mode for RJIP, we consider a luma preference mode in the YCbCr colour space. This technique is an application of image colourisation that has been successfully used in the codec R-EED [103]. Its core idea is to store more data for the luma channel Y to reconstruct the image colour accurately and fill in the colour information in the Cb and Cr channels from very sparse masks.

To this end, we distribute the total file size budget B between Y and CbCr according to a new parameter, the *luma factor* f such that $B_{\mathrm{Y}} = f \cdot B_{\mathrm{CbCr}}$. The compression pipeline remains the same as in Section 5.2: On a regular mask, we employ uniform quantisation, joint inpainting and prediction, and FSE encoding. However, we use a separate mask for the Y channel and a joint mask for the Cb and Cr channels. The respective grid sizes h and quantisation parameters q are chosen to fulfil the budget constraints. Afterwards, we perform tonal optimisation for all channels.

5.3.2 Tonal Optimisation

We do not use the random walk trial-and-error approach of RJIP described in Section 5.2 for tonal optimisation. Instead, we propose a direct, more efficient algorithm. We exploit that the truncated weights in Eq. (5.1) limit the influence of each pixel to a local area. In the following, given a current pixel value, u_i^{old} at $\boldsymbol{x}_i \in K$, we want to find u_i^{new}, such that it minimises the mean squared error (MSE).

To shorten notation, we define the value accumulation map v as the numerator of Eq. (5.1) and write $w_{i,j}$ for $w(\boldsymbol{x}_j - \boldsymbol{x}_i)$ for weights at positions $\boldsymbol{x}_j$ from a neighbourhood $\mathcal{N}_i$ relative to its centre $\boldsymbol{x}_i$. Then the new error after changing u_i^{old} to u_i^{new} is given by

$$e(u_i^{\text{new}}) = \sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \left(f_j - \frac{v_j + w_{i,j}\left(u_i^{\text{new}} - u_i^{\text{old}}\right)}{w_j} \right)^2, \tag{5.2}$$

where

$$w_j = \sum_{\boldsymbol{x}_j \in \mathcal{N}_i} w_{i,j}.$$

Since the optimal new tonal value u_i^{new} should minimise $e(\cdot)$, we solve

$$\frac{d}{du_i^{\text{new}}} e(u_i^{\text{new}}) = 0 \tag{5.3}$$

for u_i^{new} and obtain

$$u_i^{\text{new}} = \frac{\sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \frac{G_{i,j}}{w_j} \left(f_j - \frac{v_j - G_{i,j} u_i^{\text{old}}}{w_j} \right)}{\sum_{\boldsymbol{x}_j \in \mathcal{N}_i} \frac{G_{i,j}^2}{w_j^2}}. \tag{5.4}$$

Instead of testing neighbouring quantisation levels, we can now directly compute the locally optimal value. However, note that we must project these unconstrained solutions onto the set of admissible quantisation values after each computation. Iterating these steps eventually converges to a fully optimised inpainting mask.

5.4 RJIP with Vector Quantisation

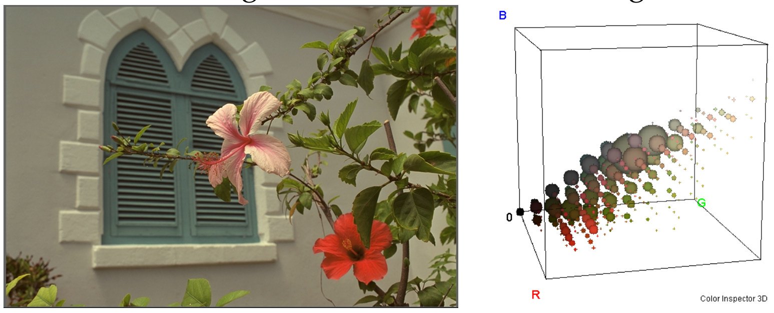
We propose the first full vector quantisation mode for inpainting-based colour image compression as an alternative to the LP mode from Section 5.2. In the following, we describe the corresponding modifications to the compression pipeline of RJIP.

5.4.1 Clustering with K-means

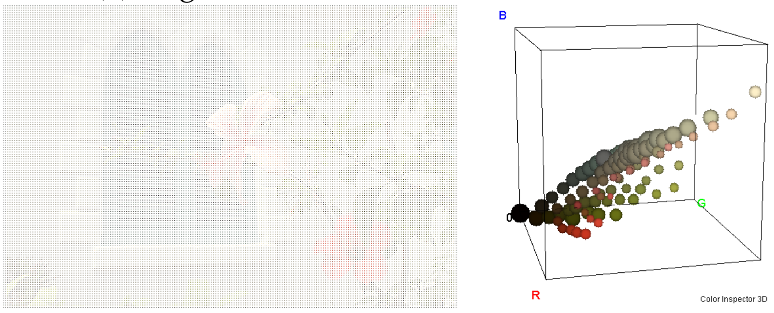
Experimental evaluations have shown that the simple k-means algorithm for clustering by Lloyd [77] is still one of the most successful techniques for vector quantisation [29]. Given an initial set of colour vectors $V := \{\boldsymbol{v}_1, ..., \boldsymbol{v}_n\}$, k-means clustering aims to partition V into k disjoint clusters $C_1, ..., C_k \subset V$ such that they minimise the cumulative squared Euclidean distance of the points in each cluster C_ℓ to the corresponding cluster centre $\boldsymbol{\mu}_\ell$.

To achieve this goal, the k-means algorithm first randomly selects k cluster centres. Alternating assignment and update steps refine this initialisation iteratively. The assignment step maps all colours to the cluster with the nearest mean value. In the subsequent update, the mean values are set to the centroids of the newly assigned clusters. Note that there are many more sophisticated initialisation strategies than the random one, e.g. k-means++ [11], histogram-based approaches [47, 51, 113, 117, 122], and iterative subdivision methods [119]. Even though the initialisation method impacts the quantisation error, we empirically determined that it does not significantly impact the final compression performance. The tonal and cluster post-processing steps described in the following paragraphs ensure we achieve good final results regardless of the initialisation.

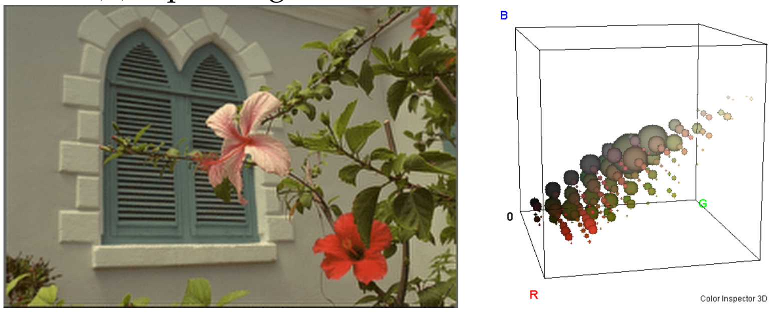
A representative histogram of an image after vector quantisation can be seen in Fig. 5.1 (c). The same figure also reveals that inpainting can approximate such a histogram well from data that is simultaneously image histogram



(a) original *kodim07*



(b) inpainting mask



(c) reconstruction

FIGURE 5.1: Colour histograms for the original image *kodim07* and the reconstructed image with Shepard inpainting. The histograms (created with Colour Inspector 3D [15]) show each bin as a ball with a radius proportional to the number of contained colours. Inpainting can reconstruct a good approximation to the original histogram from only a few quantised colours present in the inpainting mask. Here, the image is compressed to a ratio of 50:1, and the mask has only 92 different colours

sparse in the spatial domain and the colour space after coarse vector quantisation.

5.4.2 Cluster Post-Processing

We use our improved method from Section 5.3 for tonal optimisation to find optimal unconstrained values. After each iteration, we project them onto the closest cluster centre w.r.t. Euclidean distance. However, for vector quantisation, we consider an additional post-processing step.

Just as the original colour values do not necessarily yield the best inpainting result, the k-means clusters are not necessarily optimal w.r.t. reconstruction quality. As another post-processing step, we can check if moving a cluster centre to a neighbouring vector from $\{0, ..., 256\}^3$ decreases the MSE, and each such global change affects all known pixels with the same label. Thus, we optimise the quantisation codebook for final reconstruction quality and can simultaneously compensate for suboptimal initialisations of the k-means algorithm. Even though the overall quantitative gain is negligible for lower compression ratios, it becomes significant for higher ones.

5.4.3 Storing Colour Palettes and Entropy Coding

In contrast to uniform scalar quantisation, we need to store the colour palette used by the encoder. In the header, we first save the number $q \leq 256$ of quantised colours with one byte, and the cluster centres themselves with one byte per channel. The position of known data points is stored as in standard RJIP, but their value is now represented by a cluster centre index $(0, ..., q-1)$ from the stored codebook.

Finally, we apply entropy coding. Unfortunately, the joint prediction and inpainting from Section 5.2 yields unsatisfactory results in the vector case. While inpainting can still predict the colour value of neighbouring known data, the codebook labels are not tied to the distance between vectors in the colour space. Thus, an accurate prediction of the colour itself can still yield a high error in the label prediction. Therefore, we replace this step with Prediction by Partial Matching (PPM) [33].

The original version of PPM uses the linear sequence of previously encoded symbol contexts for deriving conditional probabilities for entropy

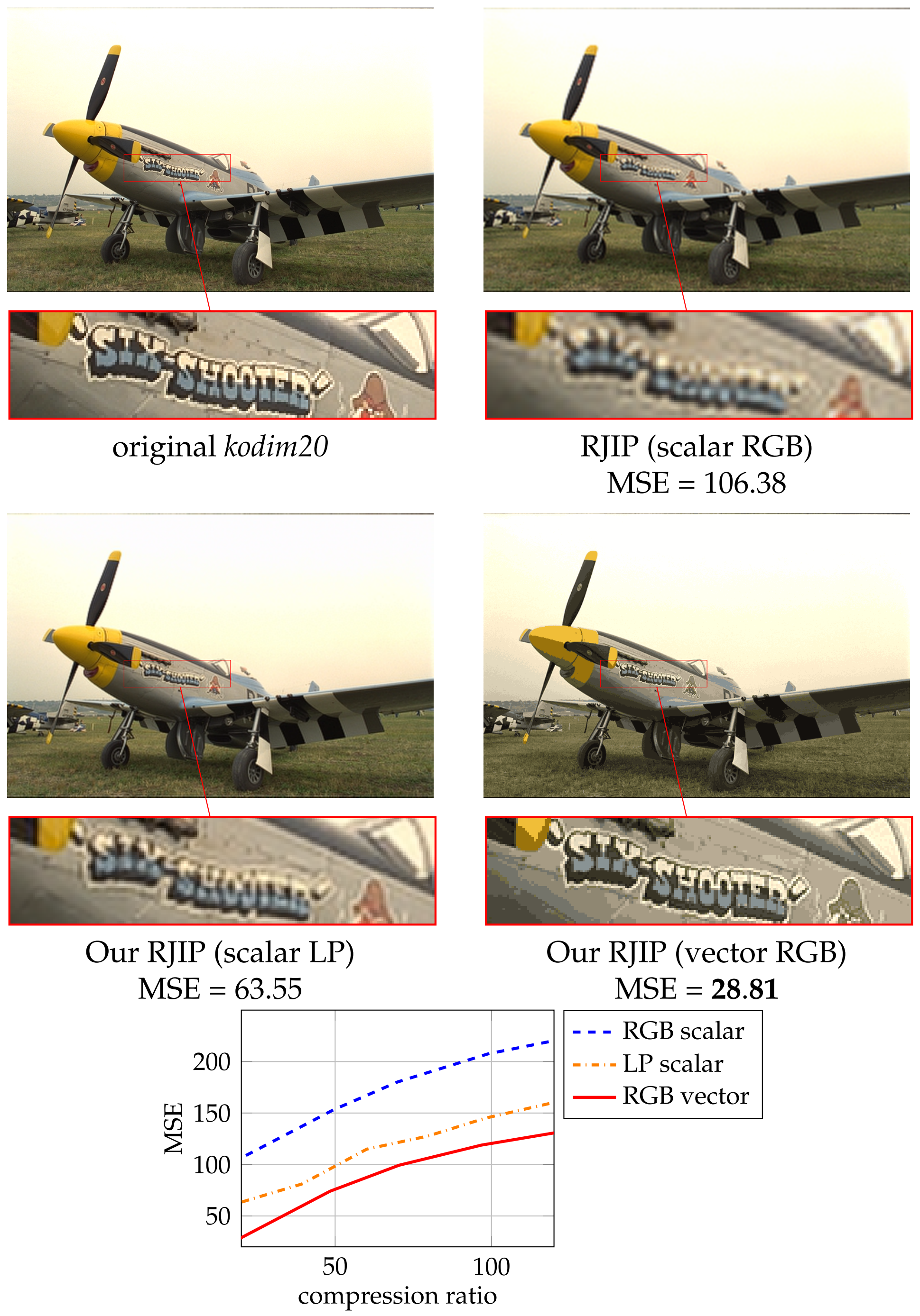


FIGURE 5.2: RJIP compression results (RGB-MSE) for *kodim20* (768 × 512 pixels) with a compression ratio of 20:1. Both our colour modes outperform RGB RJIP significantly. On images with low and moderate amount of texture, vector quantisation also outperforms LP mode considerably.

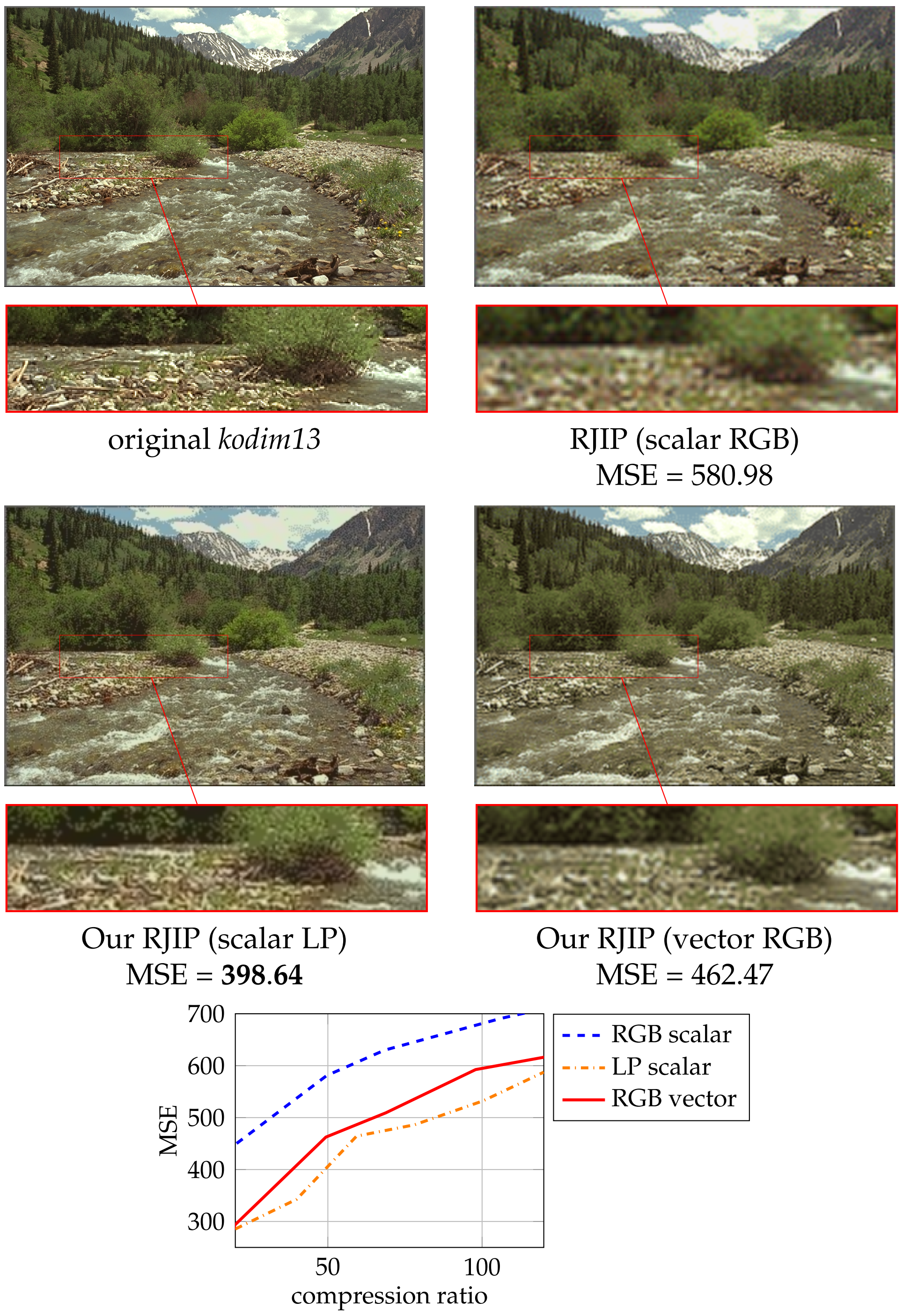


FIGURE 5.3: RJIP compression results (RGB-MSE) for *kodim13* (768 × 512 pixels) with a compression ratio of 50:1. Both our colour modes outperform RGB RJIP significantly. On images with a high amount of texture, LP mode outperforms vector quantisation considerably.

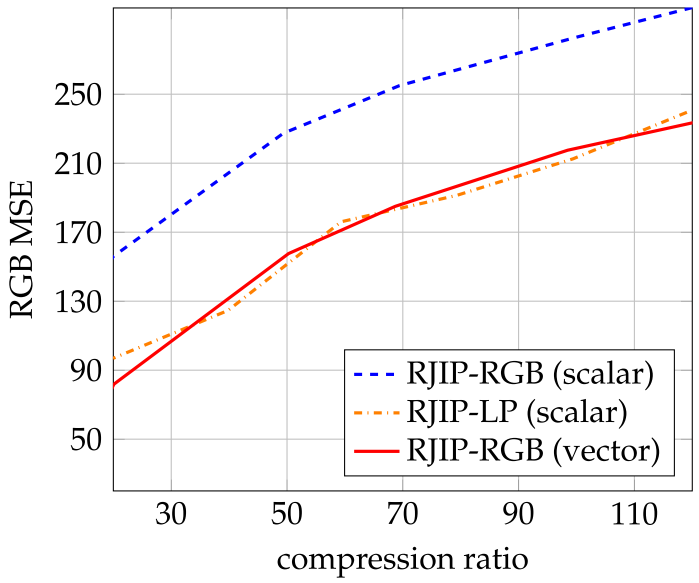


FIGURE 5.4: On the full Kodak database, both our new colour modes for RJIP outperform RGB scalar RJIP consistently and yield a similar error on average.

coding. However, this approach discards the 2-D relations of image pixels. Instead, we iterate over all mask pixels and use their three direct neighbours that have already been encoded as a 2-D context. The resulting conditional probabilities are then used for arithmetic coding.

5.5 Experiments

On the well-known Kodak image database [35], we compare the RGB scalar mode of the original RJIP [100] to our two new colour modes: RGB with vector quantisation and YCbCr luma preference mode (LP) with scalar quantisation. We use the MSE over all RGB channels as our error measure. For LP mode, we choose the luma factor $f \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ that minimises the MSE.

Fig. 5.4 shows that our new colour modes for RJIP consistently outperform the scalar RGB mode by a large margin. This also significantly improves visual quality, as illustrated by Fig. 5.2 and Fig. 5.3. On average, both colour modes yield a similar error over the whole database. However, a more detailed analysis shows that both modes have distinct advantages in different types of image content.

For images with low amounts of texture, vector quantisation is the better choice and can reduce the MSE by up to 85% compared to scalar LP, as shown in Fig. 5.2. This yields a noticeable increase in visual quality due to higher RGB mask densities. However, LP mode yields better results on heavily textured images such as the one in Fig. 5.3. Here, more data in the luma channel allows a more accurate reconstruction of the image structure.

In general, vector quantisation yields a significant speed-up compared to LP mode. Even though vector quantisation is more complex than uniform scalar quantisation due to k-means clustering, it does not require to optimise the luma factor f. This yields a runtime reduction of up to 50%.

5.6 Conclusions

Our two new colour modes for RJIP offer a significant visual and quantitative improvement over the standard RGB mode. Moreover, we assess the potential of vector quantisation for the first time. Our evaluation reveals that inpainting-based image compression can reconstruct a wide range of colours from a sparse codebook. A scalar luma preference mode can reproduce the image structure more accurately on highly textured images. In future research, we want to investigate our colour modes on data with many channels, e.g. hyperspectral imaging [8].

Chapter 6

Conclusions and Outlook

In this thesis, the goal was to systematically identify successful working principles of inpainting-based compression methods and to build new efficient codecs based on those findings. Fully optimised masks were regularly considered non-viable for end-to-end compression methods due to high storage costs despite offering the best quality for a given density. In this thesis, we explored existing compression methods dealing with sparse binary images and general data compression methods to make unconstrained mask storage more efficient. As a result, we propose two families of coders to store sparse binary images: one that compresses the image directly and the other that compresses run-lengths in Chapter 3.

From our experiments, we can provide solid recommendations for any inpainting-based coder required to store unconstrained masks. We suggest that the compression method needs to be adapted to the distribution of mask points and not the density of the mask. Nevertheless, storing an unconstrained mask over a constrained mask can only be considered if the improvement in reconstruction quality outweighs the increase in storage costs. In the future, codecs could also account for the storage costs when performing spatial optimisation. Our work helps all current inpainting-based compression methods that employ unconstrained masks.

Inpainting-based compression methods have primarily used partial differential equations (PDEs) as their inpainting method, which in most cases offer good quality. Still, they are slow and require significant numerical expertise to speed up. In this thesis, we set out to propose an alternative for PDEs which has similar reconstruction quality and is efficient and straightforward to implement. To that end, we study Shepard inpainting as a potential candidate.

We have proposed a family of image compression methods that employ Shepard inpainting in Chapter 4. We also introduce concepts such as anisotropic Shepard inpainting and direct tonal optimisation, which lead to improved results compared to the original RJIP codec, especially at lower compression ratios. We found that our codecs can be orders of magnitude faster than classical PDE-based inpainting methods and, in addition, are easy to implement.

Even though we can treat multi-channel images as many single-channel images combined into one and compress them as such, we can attain better compression performance if compression methods are tailored for multi-channel images. In this thesis, we delve into multi-channel image compression.

We present two new colour modes for RJIP that are specially tailored for RGB images in Chapter 5. These colour modes offer better visual quality and compression performance than the original RGB mode. Furthermore, we observed that inpainting could reconstruct a broader palette of colours from a stored codebook of colours obtained through vector quantisation. At the same time, the luma preference mode can reproduce image structures better.

In the future, the colour-specific modes we proposed for RJIP could be combined with the other improvements we suggested, such as anisotropic Shepard inpainting and subdivision. All upgrades for Shepard inpainting indicated in this thesis can also be applied to other inpainting methods that perform weighted averaging with slight modifications. This thesis presents Shepard inpainting as a solid alternative to classical PDE-based inpainting. Shepard inpainting has the potential to be used in many applications as is, and it could also be improved in the future if the weighting between pixels is learned and refined instead of depending on only the Euclidean distance.

Through the use of PAQ [81], existing inpainting-based codecs and those proposed in this thesis already use learning, albeit only for entropy coding. In the future, deep learning could work its way into every aspect of inpainting-based compression. A deep-learning approach for end-to-end inpainting-based codecs, where the spatial and tonal optimisation, the inpainting, and the entropy coding are learned, could be very interesting. At the very least, our findings regarding mask storage could be used in conjunction with the already existing deep-learning-based data optimisation methods [99, 105, 112] or inpainting methods [96, 99, 124, 125, 137, 138] to produce full-fledged end-to-end compression methods.

Appendix A

Bibliography

[1] T. Acar and M. Gökmen. “Image coding using weak membrane model of images”. In: *Visual Communications and Image Processing*. Ed. by A. K. Katsaggelos. Vol. 2308. Proceedings of SPIE. Bellingham: SPIE Press, 1994, pp. 1221–1230 (cit. on p. 14).

[2] R. Achanta, N. Arvanitopoulos, and S. Süsstrunk. “Extreme image completion”. In: *Proc. 42nd IEEE International Conference on Acoustics, Speech and Signal Processing*. New Orleans, LA, Mar. 2017, pp. 1333–1337 (cit. on pp. 4, 8, 12, 47, 68).

[3] R. D. Adam, P. Peter, and J. Weickert. “Denoising by Inpainting”. In: *Scale Space and Variational Methods in Computer Vision*. Ed. by F. Lauze, Y. Dong, and A. B. Dahl. Vol. 10302. Lecture Notes in Computer Science. Cham: Springer, 2017, pp. 121–132 (cit. on p. 26).

[4] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. Van Gool. “Soft-to-hard vector quantization for end-to-end learning compressible representations”. In: *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, Dec. 2017, pp. 1141–1151 (cit. on p. 21).

[5] T. Alt. “Connecting Mathematical Models for Image Processing and Neural Networks”. PhD thesis. Faculty of Mathematics and Computer Science, Saarland University, 2022 (cit. on pp. 4, 13, 48, 49).

[6] T. Alt, K. Schrader, M. Augustin, P. Peter, and J. Weickert. “Connections between numerical algorithms for PDEs and neural networks”. In: *Journal of Mathematical Imaging and Vision* 65.1 (Jan. 2023), pp. 185–208 (cit. on p. 16).

[7] T. Alt, K. Schrader, J. Weickert, P. Peter, and M. Augustin. "Designing rotationally invariant neural networks from PDEs and variational methods". In: *Research in the Mathematical Sciences* 9.3 (Sept. 2022), p. 52 (cit. on p. 16).

[8] N. Amrani, J. Serra-Sagrista, P. Peter, and J. Weickert. "Diffusion-based inpainting for coding remote-sensing data". In: *IEEE Geoscience and Remote Sensing Letters* 14.8 (Aug. 2017), pp. 1203–1207 (cit. on p. 77).

[9] S. Andris, P. Peter, R. M. K. Mohideen, J. Weickert, and S. Hoffmann. "Inpainting-based video compression in FullHD". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin, and L. Simon. Vol. 12679. Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 425–436 (cit. on pp. 22, 101).

[10] S. Andris, P. Peter, and J. Weickert. "A proof-of-concept framework for PDE-based video compression". In: *Proc. 32nd Picture Coding Symposium (PCS 2016)*. Nuremberg, Germany, Dec. 2016 (cit. on p. 22).

[11] D. Arthur and S. Vassilvitskii. "K-means++: The Advantages of Careful Seeding". In: *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. New Orleans, LA, Jan. 2007, pp. 1027–1035 (cit. on p. 71).

[12] M. Augustin, J. Weickert, and S. Andris. "Pseudodifferential Inpainting: The Missing Link Between PDE- and RBF-Based Interpolation". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by J. Lellmann, M. Burger, and J. Modersitzki. Cham: Springer, 2019, pp. 67–78 (cit. on pp. 8, 11, 22).

[13] F. Aurenhammer, R. Klein, and D. Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013 (cit. on p. 55).

[14] T. Barbu. "Segmentation-based non-texture image compression framework using anisotropic diffusion models". In: *Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science* 20.2 (2019), pp. 122–130 (cit. on p. 14).

[15] K. U. Barthel. *Color Inspector 3D*. http://rsb.info.nih.gov/ij/plugins/color-inspector.html. Last checked: September 21, 2020. 2004 (cit. on p. 72).

[16] V. Bastani, M. S. Helfroush, and K. Kasiri. "Image compression based on spatial redundancy removal and image inpainting". In: *Journal of Zhejiang University – Science C (Computers & Electronics)* 11.2 (Jan. 2010), pp. 92–100 (cit. on p. 14).

[17] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-up robust features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (June 2008), pp. 346–359 (cit. on p. 3).

[18] R. Beatson, O. Davydov, and J. Levesley. "Error bounds for anisotropic RBF interpolation". In: *Journal of Approximation Theory* 162.3 (Mar. 2010), pp. 512–527. ISSN: 0021-9045 (cit. on p. 8).

[19] Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert. "How to Choose Interpolation Data in Images". In: *SIAM Journal on Applied Mathematics* 70.1 (June 2009), pp. 333–352 (cit. on pp. 3, 13, 25).

[20] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. "Image inpainting". In: *Proc. SIGGRAPH 2000*. New Orleans, LI, July 2000, pp. 417–424 (cit. on p. 1).

[21] Å. Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, Jan. 1996. ISBN: 9781611971484 (cit. on p. 15).

[22] S. Bonettini, I. Loris, F. Porta, M. Prato, and S. Rebegoldi. "On the convergence of a linesearch based proximal-gradient method for nonconvex optimization". In: *Inverse Problems* 33.055005 (Mar. 2017) (cit. on pp. 13, 16).

[23] L. Bottou, Patrick Haffner, Paul G Howard, Patrice Simard, Yoshua Bengio, and Yann LeCun. "High quality document image compression with DjVu". In: *Journal of Electronic Imaging* 7.3 (July 1998), pp. 410–425 (cit. on pp. 18, 26, 28).

[24] T. Boutell. *PNG (Portable Network Graphics) Specification Version 1.0*. RFC 2083, Status: Informational. Jan. 1997 (cit. on pp. 17, 26, 27).

[25] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Vol. 12. Cambridge University Press, 2003 (cit. on pp. 8, 11).

[26] Q. Cai, L. Song, G. Li, and N. Ling. "Lossy and lossless intra coding performance evaluation: HEVC, H. 264/AVC, JPEG 2000 and JPEG LS". In: *Proc. 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE. 2012 (cit. on pp. 18, 19, 26, 29).

[27] S. Carlsson. "Sketch based coding of grey level images". In: *Signal Processing* 15.1 (July 1988), pp. 57–83 (cit. on pp. 7, 8, 14).

[28] G. Casciola, L. B. Montefusco, and S. Morigi. "Edge-driven image interpolation using adaptive anisotropic radial basis functions". In: *Journal of Mathematical Imaging and Vision* 36.2 (Feb. 2010), pp. 125–139 (cit. on p. 8).

[29] M. E. Celebi. "Improving the performance of k-means for color quantization". In: *Image and Vision Computing* 29.4 (Mar. 2011), pp. 260–271 (cit. on pp. 21, 71).

[30] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. "Two deterministic half-quadratic regularization algorithms for computed imaging". In: *Proc. First IEEE International Conference on Image Processing*. Vol. 2. Austin, TX, Nov. 1994, pp. 168–172 (cit. on p. 10).

[31] Y. Chen, R. Ranftl, and T. Pock. "A bi-level view of inpainting-based image compression". In: *Proc. 19th Computer Vision Winter Workshop*. Ed. by Z. Kúkelová and J. Heller. Křtiny, Czech Republic, Feb. 2014 (cit. on pp. 3, 7, 13, 15, 16, 25).

[32] V. Chizhov and J. Weickert. "Efficient Data Optimisation for Harmonic Inpainting with Finite Elements". In: *Computer Analysis of Images and Patterns*. Ed. by N. Tsapatsoulis, A. Panayides, T. Theocharides, A. Lanitis, C. Pattichis, and M. Vento. Cham: Springer, 2021, pp. 432–441 (cit. on pp. 3, 4, 10, 14, 15, 57).

[33] J. G. Cleary and I. H. Witten. "Data compression using adaptive coding and partial string matching". In: *IEEE Transactions on Communications* 32.4 (Apr. 1984), pp. 396–402 (cit. on pp. 18, 73).

[34] Y. Collet. *Finite State Entropy (FSE) Implementation*. Last checked: 2024-01-10. 2014. URL: https://github.com/Cyan4973/FiniteStateEntropy (cit. on pp. 23, 53, 69).

[35] Eastman Kodak Company. *Kodak True Color Image Suite*. Last checked: 2024-01-10. 1999. URL: http://r0k.us/graphics/kodak/ (cit. on pp. 26, 27, 57, 68, 76).

[36] V. Daropoulos, M. Augustin, and J. Weickert. "Sparse Inpainting with Smoothed Particle Hydrodynamics". In: *SIAM Journal on Imaging Sciences* 14.4 (Nov. 2021), pp. 1669–1705 (cit. on pp. 3, 8, 14, 16).

[37] L. Demaret, N. Dyn, and A. Iske. "Image compression by linear splines over adaptive triangulations". In: *Signal Processing* 86.7 (July 2006), pp. 1604–1616 (cit. on pp. 3, 14, 15, 20, 25).

[38] L. Demaret, A. Iske, and W. Khachabi. "Contextual image compression from adaptive sparse data representations". In: *Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations*. INRIA Rennes - Bretagne Atlantique. Saint Malo, France, Apr. 2009, pp. 1–6 (cit. on pp. 3, 18, 26, 33, 38, 39, 41, 43, 45).

[39] P. Deuflhard. "Cascadic conjugate gradient methods for elliptic partial differential equations: algorithm and numerical results". In: *Contemporary Mathematics* 180 (1994). Ed. by D.E.Keyes and J. Xu, pp. 29–29 (cit. on p. 22).

[40] G. Di Blasi, E. Francomano, A. Tortorici, and E. Toscano. "A smoothed particle image reconstruction method". In: *Calcolo* 48.1 (Mar. 2011), pp. 61–74 (cit. on p. 8).

[41] R. Distasi, M. Nappi, and S. Vitulano. "Image compression by B-tree triangular coding". In: *IEEE Transactions on Communications* 45.9 (Sept. 1997), pp. 1095–1100 (cit. on pp. 3, 15).

[42] J. Duchon. "Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces". In: *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique* 10.3 (Dec. 1976), pp. 5–12 (cit. on p. 7).

[43] A. A. Efros and T. K. Leung. “Texture synthesis by non-parametric sampling”. In: *Proc. Seventh IEEE International Conference on Computer Vision*. Vol. 2. Corfu, Greece, Sept. 1999, pp. 1033–1038 (cit. on p. 1).

[44] Fabrice Bellard. *BPG Specification*. Last checked: 2024-01-10. 2014. URL: http://bellard.org/bpg/bpg_spec.txt (cit. on p. 2).

[45] G. Facciolo, P. Arias, V. Caselles, and G. Sapiro. “Exemplar-based interpolation of sparsely sampled images”. In: *Energy Minimisation Methods in Computer Vision and Pattern Recognition*. Ed. by D. Cremers, Y. Boykov, A. Blake, and F. R. Schmidt. Vol. 5681. Lecture Notes in Computer Science. Berlin: Springer, 2009, pp. 331–344 (cit. on p. 8).

[46] R. Franke and G. Nielson. “Smooth interpolation of large sets of scattered data”. In: *Numerical Methods in Engineering* 15.11 (Nov. 1980), pp. 1691–1704 (cit. on p. 8).

[47] D. Freedman and P. Diaconis. “On the Histogram as a Density Estimator: L_2 Theory”. In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 57.4 (Dec. 1981), pp. 453–476 (cit. on p. 71).

[48] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. “Image compression with anisotropic diffusion”. In: *Journal of Mathematical Imaging and Vision* 31.2–3 (July 2008), pp. 255–269 (cit. on pp. 3, 7, 15, 18, 20, 33, 48, 49).

[49] J. Gautier, O. Le Meur, and C. Guillemot. “Efficient depth map compression based on lossless edge coding and diffusion”. In: *Proc. 29th Picture Coding Symposium*. Kraków, Poland, May 2012, pp. 81–84 (cit. on pp. 7, 14).

[50] A. Gersho and M. R. Gray. *Vector Quantization and Signal Compression*. Vol. 159. The Springer International Series in Engineering and Computer Science. New York: Springer, 1992 (cit. on p. 21).

[51] C. Gingles and M. E. Celebi. “Histogram-Based Method for Effective Initialization of the K-Means Clustering Algorithm”. In: *Proc. 27th International Florida Artificial Intelligence Research Society Conference*. Pensacola Beach, FL, May 2014, pp. 333–338 (cit. on p. 71).

[52] D. Hafner, P. Ochs, J. Weickert, M. Reißel, and S. Grewenig. “FSI Schemes: Fast Semi-Iterative Solvers for PDEs and Optimisation Methods”. In: *Pattern Recognition*. Ed. by B. Rosenhahn and B. Andres. Vol. 9796. Lecture Notes in Computer Science. Cham: Springer International Publishing, Aug. 2016, pp. 91–102 (cit. on p. 10).

[53] R. L. Hardy. “Multiquadric equations of topography and other irregular surfaces”. In: *Journal of Geophysical Research* 76.8 (1971), pp. 1905–1915 (cit. on pp. 8, 11).

[54] L. Hoeltgen, M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, S. Setzer, D. Johannsen, F. Neumann, and B. Doerr. “Optimising spatial and tonal data for PDE-based inpainting”. In: *Variational Methods in Imaging and Geometric Control*. Ed. by M. Bergounioux, G. Peyré, C. Schnörr, J.-P. Caillau, and T. Haberkorn. Vol. 18. Radon Series on Computational and Applied Mathematics. Berlin: De Gruyter, 2017, pp. 35–83 (cit. on pp. 14, 15).

[55] L. Hoeltgen, S. Setzer, and J. Weickert. “An optimal control approach to find sparse data for Laplace interpolation”. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Ed. by A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai. Vol. 8081. Lecture Notes in Computer Science. Berlin: Springer, 2013, pp. 151–164 (cit. on pp. 3, 13, 16, 25).

[56] L. Hoeltgen and J. Weickert. “Why Does Non-binary Mask Optimisation Work for Diffusion-Based Image Compression?” In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Ed. by Xue-Cheng Tai, Egil Bae, TonyF. Chan, and Marius Lysaker. Vol. 8932. Lecture Notes in Computer Science. Berlin: Springer, 2015, pp. 85–98 (cit. on p. 16).

[57] S. Hoffmann. “Competitive Image Compression with Linear PDEs”. PhD thesis. Faculty of Mathematics and Computer Science, Saarland University, 2016 (cit. on pp. 15, 57).

[58] S. Hoffmann, M. Mainberger, J. Weickert, and M. Puhl. “Compression of depth maps with segment-based homogeneous diffusion”. In: *Scale-Space and Variational Methods in Computer Vision*. Ed. by A. Kuijper, K. Bredies, T. Pock, and H. Bischof.

Vol. 7893. Lecture Notes in Computer Science. Berlin: Springer, 2013, pp. 319–330 (cit. on pp. 7, 14, 16, 20, 33).

[59] S. Hoffmann, G. Plonka, and J. Weickert. "Discrete Green's functions for harmonic and biharmonic inpainting with sparse atoms". In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Ed. by X.-C. Tai, E. Bae, T. F. Chan, and M. Lysaker. Vol. 8932. Lecture Notes in Computer Science. Berlin: Springer, 2015, pp. 169–182 (cit. on pp. 3, 4, 10).

[60] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge. "The emerging JBIG2 standard". In: *IEEE Transactions on Circuits, Systems and Video Technology* 8.7 (Nov. 1998), pp. 838–848 (cit. on pp. 18, 26, 29).

[61] D. A. Huffman. "A method for the construction of minimum redundancy codes". In: *Proceedings of the IRE* 40.9 (Sept. 1952), pp. 1098–1101 (cit. on p. 17).

[62] T. Iijima. "Basic theory of pattern observation". In: *Papers of Technical Group on Automata and Automatic Control*. In Japanese. Kyoto, Japan: IECE, 1959 (cit. on p. 26).

[63] T. Iijima. "Basic theory on normalization of pattern (in case of typical one-dimensional pattern)". In: *Bulletin of the Electrotechnical Laboratory* 26 (Jan. 1962). In Japanese, pp. 368–388 (cit. on pp. 7, 8, 9, 51, 57).

[64] Joint Bi-level Image Experts Group. *Information Technology – Progressive Lossy/Lossless Coding of Bi-level Images*. Standard. International Organization for Standardization, 1993 (cit. on pp. 18, 22, 26, 28).

[65] F. Jost, P. Peter, and J. Weickert. "Compressing flow fields with edge-aware homogeneous diffusion inpainting". In: *Proc. 45th International Conference on Acoustics, Speech, and Signal Processing*. Barcelona, Spain, May 2020 (cit. on pp. 7, 14, 15, 22).

[66] F. Jost, P. Peter, and J. Weickert. "Compressing piecewise smooth images with the Mumford–Shah cartoon model". In: *Proc. 28th European Signal Processing Conference*. Amsterdam, Netherlands, Jan. 2021, pp. 511–515 (cit. on pp. 2, 15, 16, 22, 48, 66).

[67] I. Jumakulyyev and T. Schulz. “Fourth-order anisotropic diffusion for inpainting and image compression”. In: *Anisotropy Across Fields and Scales*. Ed. by E. Özarslan, T. Schulz, E. Zhang, and A. Fuster. Cham: Springer, 2021, pp. 99–124 (cit. on p. 7).

[68] E. M. Kalmoun and M. M. S. Nasser. “Harmonic image inpainting using the charge simulation method”. In: *Pattern Analysis and Applications* 25.4 (Apr. 2022), pp. 795–806 (cit. on pp. 3, 4, 10).

[69] N. Kämper and J. Weickert. “Domain Decomposition Algorithms for Real-Time Homogeneous Diffusion Inpainting in 4K”. In: *Proc. 2022 IEEE International Conference on Acoustics, Speech and Signal Processing*. May 2022, pp. 1680–1684 (cit. on pp. 3, 4, 10).

[70] L. Karos, P. Bheed, P. Peter, and J. Weickert. “Optimising data for exemplar-based inpainting”. In: *Advanced Concepts for Intelligent Vision Systems*. Ed. by J. Blanc-Talon, D. Helbert, W. Philips, D. Popescu, and P. Scheunders. Vol. 11182. Lecture Notes in Computer Science. Cham: Springer, 2018, pp. 547–558 (cit. on pp. 3, 25).

[71] H. Knutsson and C.-F. Westin. “Normalized and differential convolution”. In: *Proc. 1993 IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY, June 1993, pp. 515–523 (cit. on p. 8).

[72] H. Köstler, M. Stürmer, C. Freundl, and U. Rüde. *PDE Based Video Compression in Real Time*. Tech. rep. 07-11. Germany: Lehrstuhl für Informatik 10, University Erlangen–Nürnberg, 2007 (cit. on pp. 3, 4, 10).

[73] P. Peter L. Hoeltgen and M. Breuß. “Clustering-based quantisation for PDE-based image compression”. In: *Signal, Image and Video Processing* 12.3 (Mar. 2018), pp. 411–419 (cit. on p. 21).

[74] Y. Li, M. Sjostrom, U. Jennehag, and R. Olsson. “A scalable coding approach for high quality depth image compression.” In: *Proc. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*. Zurich, Switzerland, Oct. 2012, pp. 1–4 (cit. on p. 14).

[75] Z. Li, X. Zhang, R. Zhu, Z. Zhang, and Z. Weng. "Integrating data-to-data correlation into inverse distance weighting". In: *Computational Geosciences* 24 (Nov. 2019), pp. 203–216 (cit. on p. 8).

[76] C. Livada, I. Galić, and B. Zovko-Cihlar. "EEDC image compression using Burrows-Wheeler data modeling". In: *Proc. Electronics in Marine*. Zadar, Croatia, Sept. 2012, pp. 1–5 (cit. on p. 15).

[77] S. P. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137 (cit. on pp. 21, 71).

[78] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proc. 7th IEEE International Conference on Computer Vision*. Kerkyra, Greece, Sept. 1999, pp. 1150–1157 (cit. on p. 3).

[79] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. "RePaint: Inpainting using Denoising Diffusion Probabilistic Models". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2022, pp. 11451–11461 (cit. on p. 8).

[80] F. Magoulès, L. A. Diago, and I. Hagiwara. "A two-level iterative method for image reconstruction with radial basis functions". In: *JSME International Journal Series C, Mechanical Systems, Machine Elements and Manufacturing* 48.2 (2005), pp. 149–158 (cit. on p. 8).

[81] M. Mahoney. *Adaptive weighing of context models for lossless data compression*. Tech. rep. CS-2005-16. Melbourne, FL: Florida Institute of Technology, Dec. 2005 (cit. on pp. 22, 26, 33, 34, 80).

[82] M. Mahoney. *LPAQ*. Last checked: 2024-01-10. 2007. URL: `http://mattmahoney.net/dc/#lpaq` (cit. on pp. 19, 26, 56).

[83] M. Mahoney. *The PAQ1 data compression program*. Tech. rep. Melbourne, FL: Florida Institute of Technology, Mar. 2002 (cit. on p. 19).

[84] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. "Edge-based compression of cartoon-like images with homogeneous diffusion". In: *Pattern Recognition* 44.9 (Sept. 2011), pp. 1859–1873 (cit. on pp. 3, 4, 7, 10, 14, 22, 28).

[85] 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: *Scale Space and Variational Methods in Computer Vision*. Ed. by A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein. Vol. 6667. Lecture Notes in Computer Science. Berlin: Springer, 2012, pp. 26–37 (cit. on pp. 3, 14, 15, 25, 26).

[86] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proc. Eigth International Conference on Computer Vision*. Vancouver, Canada, July 2001, pp. 416–423 (cit. on pp. 57, 62).

[87] D. Marwood, P. Massimino, M. Covell, and S. Baluja. "Representing images in 200 bytes: Compression via triangulation". In: *Proc. 25th IEEE International Conference on Image Processing*. Athens, Greece, Oct. 2018, pp. 405–409 (cit. on pp. 3, 16, 18, 20, 26, 33, 39).

[88] S. Masnou and J.-M. Morel. "Level lines based disocclusion". In: *Proc. 1998 IEEE International Conference on Image Processing*. Vol. 3. Chicago, IL, Oct. 1998, pp. 259–263 (cit. on p. 1).

[89] R. M. K. Mohideen, T. Alt, P. Peter, and J. Weickert. *Image Compression with Isotropic and Anisotropic Shepard Inpainting*. arXiv:2406.06247 [eess.IV]. June 2024 (cit. on pp. 48, 101).

[90] R. M. K. Mohideen, P. Peter, T. Alt, J. Weickert, and A. Scheer. *Compressing Colour Images with Joint Inpainting and Prediction*. arXiv:2010.09866 [eess.IV]. Oct. 2020 (cit. on pp. 67, 101).

[91] R. M. K. Mohideen, P. Peter, and J. Weickert. "A Systematic Evaluation of Coding Strategies for Sparse Binary Images". In: *Signal Processing: Image Communication* 99.116424 (Nov. 2021) (cit. on pp. 14, 25, 101).

[92] K. W. Morton and L. M. Mayers. *Numerical Solution of Partial Differential Equations*. Second. Cambridge, UK: Cambridge University Press, 2005 (cit. on p. 10).

[93] D. Mumford and J. Shah. “Boundary detection by minimizing functionals”. In: *Proc. First IEEE Conference on Computer Vision and Pattern Recognition*. San Francisco, CA, June 1985, pp. 22–26 (cit. on p. 22).

[94] R. Nahme. “Inertial proximal algorithms in diffusion-based image compression”. Master Thesis. Department of Mathematics, University of Göttingen, Germany, 2015 (cit. on pp. 13, 16).

[95] P. Ochs, Y. Chen, T. Brox, and T. Pock. “iPiano: Inertial proximal algorithm for nonconvex optimization”. In: *SIAM Journal on Applied Mathematics* 7.2 (June 2014), pp. 1388–1419 (cit. on pp. 13, 16).

[96] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. “Context Encoders: Feature Learning by Inpainting”. In: *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, June 2016, pp. 2536–2544 (cit. on pp. 8, 81).

[97] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. New York: Springer, 1992 (cit. on pp. 2, 4, 17, 21, 67).

[98] P. Perona and J. Malik. “Scale space and edge detection using anisotropic diffusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (July 1990), pp. 629–639 (cit. on pp. 8, 50).

[99] P. Peter. “A Wasserstein GAN for Joint Learning of Inpainting and its Spatial Optimisation”. In: *Image and Video Technology*. Ed. by H. Wang, W. Lin, P. Manoranjan, G. Xiao, K. P. Chan, X. Wang, G. Ping, and H. Jiang. Vol. 13763. lncs. Cham: Springer International Publishing, Apr. 2023, pp. 132–145 (cit. on pp. 8, 16, 81).

[100] P. Peter. “Fast Inpainting-Based Compression: Combining Shepard Interpolation with Joint Inpainting and Prediction”. In: *Proc. 26th IEEE International Conference on Image Processing*. Taipei, Taiwan, Sept. 2019, pp. 3557–3561 (cit. on pp. 3, 8, 14, 20, 48, 52, 66, 67, 68, 76).

[101] P. Peter, J. Contelly, and J. Weickert. "Compressing Audio Signals with Inpainting-Based Sparsification". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by J. Lellmann, M. Burger, and J. Modersitzki. Cham: Springer International Publishing, 2019, pp. 92–103 (cit. on p. 16).

[102] P. Peter, S. Hoffmann, F. Nedwed, L. Hoeltgen, and J. Weickert. "Evaluating the true potential of diffusion-based inpainting in a compression context". In: *Signal Processing: Image Communication* 46 (Aug. 2016), pp. 40–53 (cit. on pp. 3, 7, 15, 16, 20, 33, 60).

[103] P. Peter, L. Kaufhold, and J. Weickert. "Turning diffusion-based image colorization into efficient color compression". In: *IEEE Transactions on Image Processing* 26.2 (Nov. 2016), pp. 860–869 (cit. on pp. 7, 15, 16, 20, 22, 33, 67, 69).

[104] P. Peter, C. Schmaltz, N. Mach, M. Mainberger, and J. Weickert. "Beyond Pure Quality: Progressive Modes, Region of Interest Coding, and Real Time Video Decoding for PDE-based Image Compression." In: *Journal of Visual Communication and Image Representation* 31.4 (Aug. 2015), pp. 253–265 (cit. on p. 4).

[105] P. Peter, K. Schrader, T. Alt, and J. Weickert. "Deep spatial and tonal data optimisation for homogeneous diffusion inpainting". In: *Pattern Analysis and Applications* 26.4 (Apr. 2023), pp. 1585–1600 (cit. on pp. 13, 16, 81).

[106] P. Peter and J. Weickert. "Compressing images with diffusion- and exemplar-based inpainting". In: *Scale-Space and Variational Methods in Computer Vision*. Ed. by J.-F. Aujol, M. Nikolova, and N. Papadakis. Vol. 9087. Lecture Notes in Computer Science. Berlin: Springer, 2015, pp. 154–165 (cit. on p. 8).

[107] R. J. Renka. "Multivariate interpolation of large sets of scattered data". In: *ACM Transactions on Mathematical Software* 14.2 (June 1988), pp. 139–148 (cit. on p. 8).

[108] T. Roosendaal. "Sintel". In: *ACM SIGGRAPH 2011 Computer Animation Festival*. New York, NY, USA: Association for Computing Machinery, 2011, p. 71. ISBN: 9781450309660 (cit. on p. 56).

[109] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Philadelphia: SIAM, 2003 (cit. on pp. 26, 31).

[110] K. Sayood. "Arithmetic Coding". In: *Introduction to Data Compression*. fifth. Elsevier, 2018. Chap. 4, pp. 89–130 (cit. on p. 17).

[111] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert, and A. Bruhn. "Understanding, Optimising, and Extending Data Compression with Anisotropic Diffusion". In: *International Journal of Computer Vision* 108.3 (July 2014), pp. 222–240 (cit. on pp. 2, 3, 7, 15, 16, 18, 20, 22, 33, 47, 48, 54).

[112] K. Schrader, P. Peter, N. Kämper, and J. Weickert. "Efficient Neural Generation of 4K Masks for Homogeneous Diffusion Inpainting". In: *Scale Space and Variational Methods in Computer Vision*. Ed. by L. Calatroni, M. Donatelli, S. Morigi, M. Prato, and M. Santacesaria. Vol. 14009. Lecture Notes in Computer Science. Cham: Springer International Publishing, May 2023, pp. 16–28 (cit. on pp. 16, 81).

[113] D. W. Scott. "On Optimal and Data-Based Histograms". In: *Biometrika* 66.3 (Dec. 1979), pp. 605–610 (cit. on p. 71).

[114] J. Seward. *bzip2 and libbzip2, Version 1.0.5, A Program and Library for Data Compression*. Last checked: 2024-01-10. 2008. URL: `http://www.bzip.org/.` (cit. on p. 22).

[115] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Urbana: University of Illinois Press, 1949 (cit. on pp. 17, 21).

[116] D. Shepard. "A two-dimensional interpolation function for irregularly-spaced data". In: *Proc. 23rd ACM National Conference*. Las Vegas, NV, Aug. 1968, pp. 517–524 (cit. on pp. 4, 8, 11, 26, 47, 68).

[117] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Boca Raton: Chapman & Hall, 1986 (cit. on p. 71).

[118] K. Somasundaram and M. S. Rani. "Eigen Value based k-means clustering for image compression". In: *International Journal of Applied Information Systems* 3.7 (Aug. 2012), pp. 21–24 (cit. on p. 21).

[119] T. Su and J. Dy. "A Deterministic Method for Initializing K-Means Clustering". In: *Proc. 16th IEEE International Conference on Tools with Artificial Intelligence*. Boca Raton, FL, Nov. 2004, pp. 784–786 (cit. on p. 71).

[120] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand. “Overview of the high efficiency video coding (HEVC) standard”. In: *IEEE Transactions on Circuits, Systems and Video Technology* 22.12 (Sept. 2012), pp. 1649–1668 (cit. on pp. 18, 29).

[121] D. S. Taubman and M. W. Marcellin, eds. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Boston: Kluwer, 2002 (cit. on pp. 2, 4, 67).

[122] G. R. Terrell. “The Maximal Smoothing Principle in Density Estimation”. In: *Journal of the American Statistical Association* 85.410 (June 1990), pp. 470–477 (cit. on p. 71).

[123] J. A. Tómasson, P. Ochs, and J. Weickert. “AFSI: Adaptive Restart for Fast Semi-Iterative Schemes for Convex Optimisation”. In: *Pattern Recognition*. Ed. by T. Brox, A. Bruhn, and M. Fritz. Vol. 11269. Lecture Notes in Computer Science. Cham: Springer International Publishing, Feb. 2019, pp. 669–681 (cit. on p. 10).

[124] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Deep Image Prior”. In: *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, June 2018, pp. 9446–9454 (cit. on pp. 8, 81).

[125] D. Vašata, T. Halama, and M. Friedjungová. “Image Inpainting Using Wasserstein Generative Adversarial Imputation Network”. In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Ed. by I. Farkaš, P. Masulli, S. Otte, and S. Wermter. Vol. 12892. Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 575–586 (cit. on pp. 8, 81).

[126] N. Venkateswaran and Y. V. Ramana Rao. “K-means clustering based image compression in wavelet domain”. In: *Information Technology Journal* 6.1 (Jan. 2007), pp. 148–153 (cit. on p. 21).

[127] L. Wang, B.-S. Hua, and X. Li. “Adaptive Energy Diffusion for Blind Inverse Halftoning”. In: *Advances in Multimedia Information Processing - PCM 2010*. Ed. by G. Qiu, K. M. Lam, H. Kiya, X.-Y. Xue, C.-C. J. Kuo, and M. S. Lew. Vol. 6297. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, Sept. 2010, pp. 470–480 (cit. on p. 55).

[128] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612 (cit. on p. 61).

[129] J. Weickert. *Anisotropic Diffusion in Image Processing*. Stuttgart: Teubner, 1998 (cit. on pp. 8, 10).

[130] J. Weickert. "Theoretical foundations of anisotropic diffusion in image processing". In: *Computing Supplement* 11 (1996), pp. 221–236 (cit. on pp. 7, 9, 20, 22, 49).

[131] J. Weickert, S. Grewenig, C. Schroers, and A. Bruhn. "Cyclic schemes for PDE-based image analysis". In: *International Journal of Computer Vision* 118.3 (July 2016), pp. 275–299 (cit. on p. 10).

[132] J. Weickert and M. Welk. "Tensor field interpolation with PDEs". In: *Visualization and Processing of Tensor Fields*. Ed. by J. Weickert and H. Hagen. Berlin: Springer, 2006, pp. 315–325 (cit. on pp. 7, 8, 9, 20, 22, 49).

[133] H. Wendland. *Scattered Data Approximation*. Cambridge, UK: Cambridge University Press, 2005 (cit. on pp. 8, 11).

[134] H. Werner. "Studies on contour". In: *The American Journal of Psychology* 47.1 (Jan. 1935), pp. 40–64 (cit. on p. 2).

[135] I. H. Witten, R. M. Neal, and J. G. Cleary. "Arithmetic coding for data compression". In: *Communications of the ACM* 30.6 (June 1987), pp. 520–540 (cit. on pp. 18, 22, 54).

[136] 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*. Sanya, China, Apr. 2009, pp. 816–820 (cit. on p. 14).

[137] J. Xie, L. Xu, and E. Chen. "Image Denoising and Inpainting with Deep Neural Networks". In: *Proc. 26th International Conference on Neural Information Processing Systems*. Ed. by P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Advances in Neural Information Processing Systems. Lake Tahoe, NV, Dec. 2012, pp. 350–358 (cit. on pp. 8, 81).

[138] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. "High-Resolution Image Inpainting Using Multi-Scale Neural Patch Synthesis". In: *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, July 2017, pp. 6721–6729 (cit. on pp. 8, 81).

[139] C. Zhao and M. Du. "Image Compression based on PDEs". In: *Proc. 2011 International Conference of Computer Science and Network Technology*. Vol. 3. Harbin, China, Dec. 2011, pp. 1768–1771 (cit. on p. 14).

[140] Q. Zhou, H. Yao, F. Cao, and Y.-C. Hu. "Efficient image compression based on side match vector quantization and digital inpainting". In: *Journal of Real-Time Image Processing* 16.3 (June 2019), pp. 799–810 (cit. on p. 21).

[141] J. Ziv and A. Lempel. "An universal algorithm for sequential data compression". In: *IEEE Transactions on Information Theory* 23.3 (May 1977), p. 337 (cit. on p. 28).

Appendix B

Own Publications

Journal Publications

R. M. K. Mohideen, P. Peter, and J. Weickert. “A Systematic Evaluation of Coding Strategies for Sparse Binary Images”. In: *Signal Processing: Image Communication* 99.116424 (Nov. 2021)

Conference Publications

S. Andris, P. Peter, R. M. K. Mohideen, J. Weickert, and S. Hoffmann. “Inpainting-based video compression in FullHD”. in: *Scale Space and Variational Methods in Computer Vision*. Ed. by A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin, and L. Simon. Vol. 12679. Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 425–436

Technical Reports

R. M. K. Mohideen, T. Alt, P. Peter, and J. Weickert. *Image Compression with Isotropic and Anisotropic Shepard Inpainting*. arXiv:2406.06247 [eess.IV]. June 2024

R. M. K. Mohideen, P. Peter, T. Alt, J. Weickert, and A. Scheer. *Compressing Colour Images with Joint Inpainting and Prediction*. arXiv:2010.09866 [eess.IV]. Oct. 2020

Appendix C

Performance comparisons of the Shepard family of codecs on the Kodak database

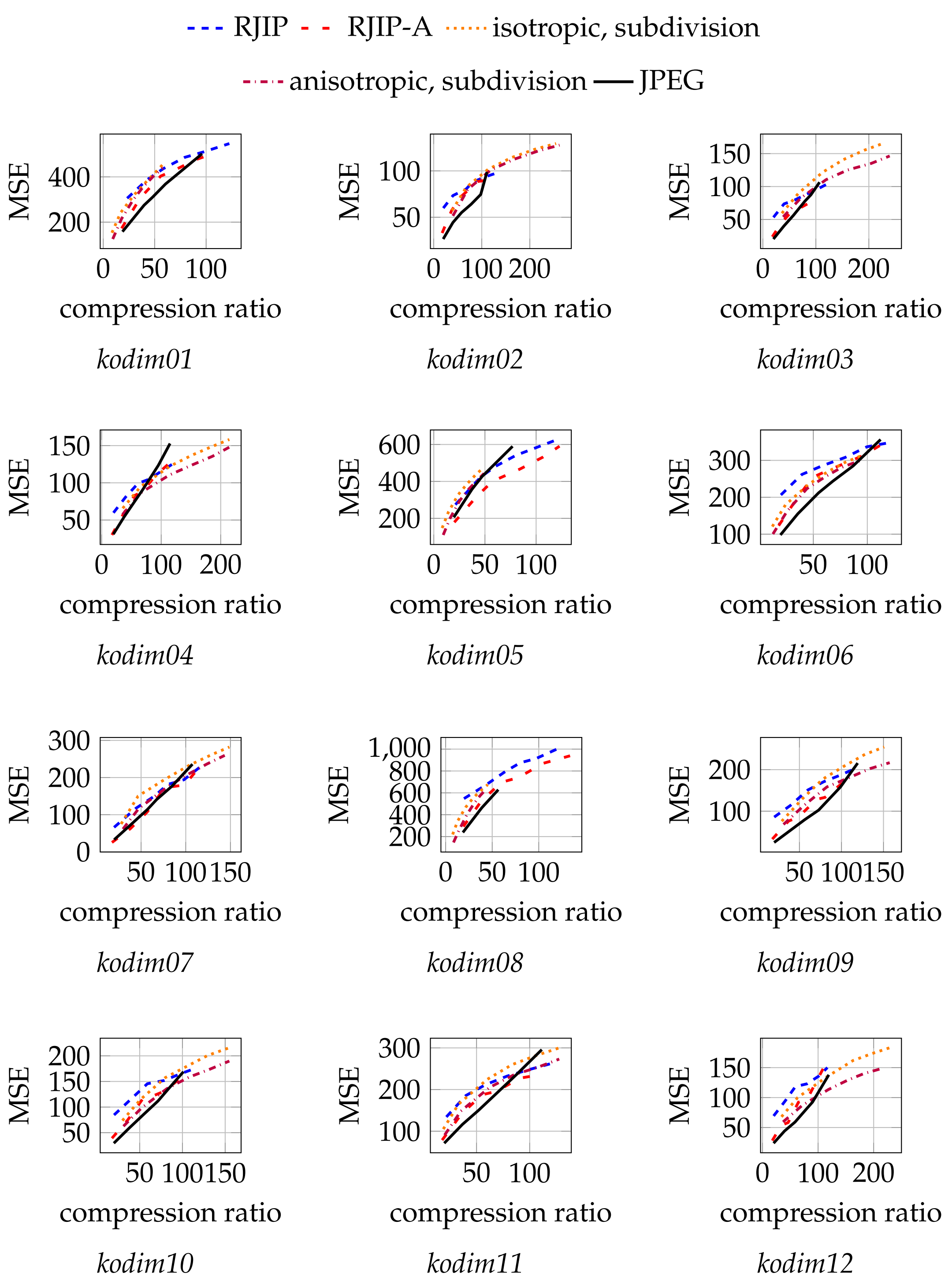


FIGURE C.1: We extend the results for our experiments in Section 4.4. We present the rate-distortion results for images 1 through 12 in the Kodak dataset.

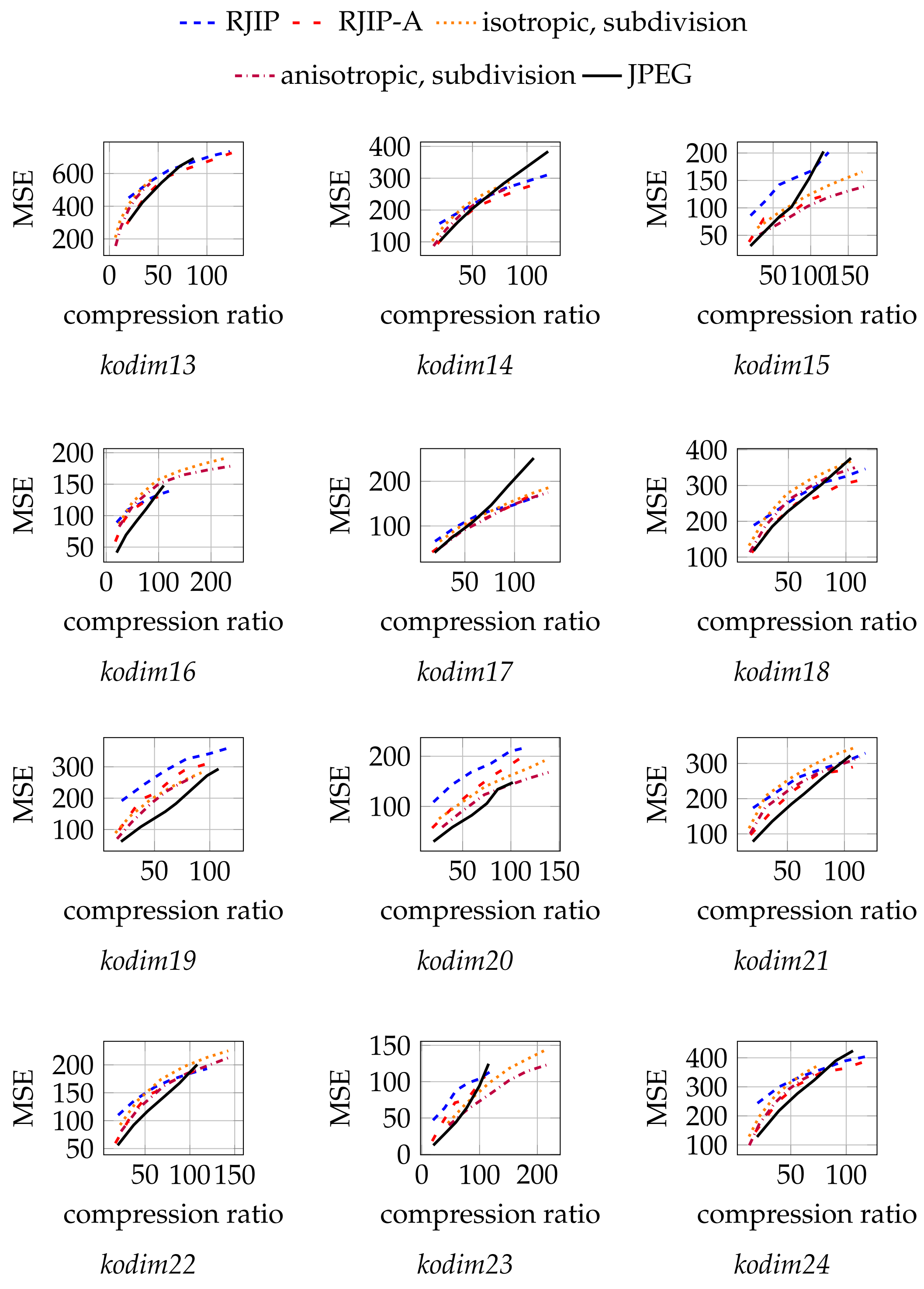


FIGURE C.2: We extend the results for our experiments in Section 4.4. We present the rate-distortion results for images 13 through 24 in the Kodak dataset.

List of Figures

List of Tables