Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 329

Understanding, Optimising, and Extending Data Compression with Anisotropic Diffusion

Christian Schmaltz, Pascal Peter, Markus Mainberger, Franziska Ebel, Joachim Weickert and Andrés Bruhn

Saarbrücken 2013

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 329
submitted: March 25, 2013

Understanding, Optimising, and Extending Data Compression with Anisotropic Diffusion

Christian Schmaltz
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.7, 66041 Saarbrücken, Germany
`schmaltz@mia.uni-saarland.de`

Pascal Peter
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.7, 66041 Saarbrücken, Germany
`peter@mia.uni-saarland.de`

Markus Mainberger
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.7, 66041 Saarbrücken, Germany
`mainberger@mia.uni-saarland.de`

Franziska Ebel
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.7, 66041 Saarbrücken, Germany
`EbelFranziska@gmail.com`

Joachim Weickert
Saarland University
Department of Mathematics and Computer Science
Mathematical Image Analysis Group
Campus E1.7, 66041 Saarbrücken, Germany
`weickert@mia.uni-saarland.de`

Andrés Bruhn
University of Stuttgart
Institute for Visualization and Interactive Systems
Intelligent Systems Group
Universitätsstraße 38, 70569 Stuttgart, Germany
`bruhn@vis.uni-stuttgart.de`

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax:	+ 49 681 302 4443
e-Mail:	preprint@math.uni-sb.de
WWW:	http://www.math.uni-sb.de/

Understanding, Optimising, and Extending Data Compression with Anisotropic Diffusion

Christian Schmaltz Pascal Peter Markus Mainberger
Franziska Ebel Joachim Weickert Andrés Bruhn

March 25, 2013

Abstract

Galić et al. [33] have shown that compression based on edge-enhancing anisotropic diffusion (EED) can outperform the quality of JPEG for medium to high compression ratios when the interpolation points are chosen as vertices of an adaptive triangulation. However, the reasons for the good performance of EED remained unclear, and they could not outperform the more advanced JPEG 2000. The goals of the present paper are threefold: Firstly, we investigate the compression qualities of various partial differential equations. This sheds light on the favourable properties of EED in the context of image compression. Secondly, we demonstrate that it is even possible to beat the quality of JPEG 2000 with EED if one uses specific subdivisions on rectangles and several important optimisations. These amendments include improved entropy coding, brightness and diffusivity optimisation, and interpolation swapping. Thirdly, we demonstrate how to extend our approach to 3-D and shape data. Experiments on classical test images and 3-D medical data illustrate the high potential of our approach.

1 Introduction

As the number and resolution of images is constantly increasing, image compression with high compression rates is becoming more and more important. In this context, in particular lossy image compression algorithms are of interest, since they achieve much higher compressions rates than their lossless counterparts. Among the most popular lossy image compression algorithms are JPEG [53], which uses a discrete cosine transform (DCT), and its successor JPEG 2000 [63], which is based on biorthogonal wavelets.

While these so-called transform-based methods mark the quasi-standards in modern applications, interesting alternatives have recently been proposed that proceed in a completely different way: They are based on *partial differential equations* (*PDE*s). The interpolation qualities of PDEs have become evident by an axiomatic analysis [20], by applying them to image inpainting [51, 10, 22, 12, 66, 11] and by utilising them for upsampling digital images [49, 7, 8, 3, 73, 58]. Extending this to image compression drives inpainting to the extreme: Only a small set of *specifically selected* pixels is stored, while the remaining image is reconstructed using the filling-in effect of PDE-based interpolation. This idea has been pursued in [32], and has later been extended with different amendments in [33]. By encoding pixel locations in a binary tree that arises from an adaptive triangulation [28], their algorithm can attain a quality that lies between JPEG and JPEG 2000 for medium to high compression ratios.

Both in [32] and [33], a specific anisotropic diffusion process has been used that is called *edge-enhancing anisotropic diffusion* (*EED*, [69]). Although the *interpolation* qualities of EED have been compared to other partial differential equations in [32] and [33], the real reason for the favourable performance of EED in the context of *compression* has not been investigated: All comparisons have been performed on a *fixed, preselected* set of pixels. This set has been extracted by randomly choosing a specified percentage of pixels. Therefore, one has evaluated the (scattered data) interpolation qualities. However, the theoretical results in [9] for image inpainting with homogeneous diffusion demonstrate that it is of crucial importance to choose an *optimised* set of interpolation points for compression. This optimisation step is the fundamental difference between PDE-based interpolation and PDE-based compression. Thus, one should also evaluate the compression qualities of the different PDEs for individually adapted sets of interpolation points. This has not been investigated so far.

An additional problem is that all image compression algorithms described above are only available for grey-valued 2-D images, while different image types such as 3-D images or binary shapes are used in important real world applications such as video coding, medical imaging [15], or geological applications.

For 3-D images, one possibility is to regard one dimension as time and employ standard video compression algorithms. However, such algorithms typically introduce a bias between successive frames, which is often undesired. Therefore, medical image standards like DICOM [52] use 2-D compression methods such as JPEG 2000. For this setting, we propose a natural extension of our compression algorithm which allows to compress 3-D data inherently, i.e. without preferring a certain direction.

Our Contribution. The goal of the present paper is threefold, namely to evaluate inpainting operators within a compression framework, to introduce a PDE-based compression algorithm that can outperform JPEG 2000, and to extend this ap-

proach to shape coding and the 3-D setting.
To this end, we first investigate a number of different PDEs in some carefully selected examples that illustrate the advantages and drawbacks of different inpainting algorithms based on partial differential equations. By comparing homogeneous diffusion, isotropic nonlinear diffusion, anisotropic nonlinear diffusion, absolute minimal Lipschitz extension, biharmonic smoothing, triharmonic smoothing, as well as PDE-based inpainting operators introduced by Tschumperlé [65], and by Bornemann and März [11], we identify EED as a particularly useful PDE for image compression. Experiments are presented that provide insights into the particular qualities that distinguish EED from other PDEs. Thus, our paper puts EED-based image compression on a more solid foundation.
Secondly, we show that it is even possible to exceed the quality of JPEG 2000 with an EED-based compression algorithm if several carefully optimised concepts are used that are not considered in [33]: First of all, we replace the adaptive triangulation by a subdivision into a rectangular structure and evaluate different point patterns on this adaptive structure. Furthermore, we use an improved entropy encoding of the stored brightness values, an optimisation of the contrast parameter within the diffusion process, and a swapping of the role of interpolation points and interpolation domain in the decoding step. The resulting novel codec that uses EED within a *rectangular* subdivision is called R-EED. We further validate our result from the first part of the paper by comparing several inpainting operators in our compression framework.
Finally, we carry over the introduced concepts to novel settings by performing shape-encoding using quadrupoles, and by introducing a novel codec for 3-D data that combines *cuboidal* subdivision with 3-D inpainting.
Organisation of the Paper. Our paper is structured as follows: Section 2 explains how to interpolate an image from single points and sketches several partial differential equations that can be used for image interpolation. In Section 3, we evaluate these methods and present a number of experiments that illustrate why EED is well suited for compressing images. Section 4 introduces our complete 2-D image compression framework, which is evaluated in Section 5. Section 6 finally extends our framework to shape coding, as well as to coding of 3-D data. The paper is concluded with a summary in Section 7.
Related Work. Let us now briefly mention some related papers that have not been discussed so far.
In the context of image compression, PDEs and related variational techniques have mainly been used as a preprocessing step before coding images or videos (see e.g. [67, 40]) or as a postprocessing tool for removing coding artifacts (see e.g. [31, 2]).
The papers [32, 33] as well as our present work differs from these strategies by the fact that we use a PDE *within* encoding and decoding rather than applying it

before encoding or after decoding. In that sense they are more related to [23] who apply total variation regularisation in order to modify the coefficients in a wavelet decomposition to reduce oscillatory artifacts.

Sometimes PDE-based interpolation strategies have been tailored to specific data sets such as surface data in digital elevation maps [30, 60, 76]. Moreover, some variational L^1 minimisation ideas play an important role in recent compressed sensing concepts [18].

The usefulness of inpainting concepts for image compression is studied in several papers, where structure and texture inpainting ideas have been integrated into standard codecs such as JPEG [44, 56, 77].

With respect to its intention to reconstruct an image from a small set of characteristic data, our paper has some relations to publications where edge information is used to represent the main image content. This has been done in many different formulations [78, 19, 36, 50, 1, 5, 26, 29, 75, 48]. Methods of this type can be seen as representatives of second-generation coding approaches that exploit perceptually relevant features such as edge contours [42].

An alternative way to represent signals and images by a sparse set of significant points consists of reconstructions from top points in scale-space, as has been investigated by [38] and [39]. More general discussions on how to reconstruct an image from a suitable set of feature points and their derivatives (local jet) have been presented by [43]. Impressive global reconstructions of natural images by means of the local jet structure are reported in a classical technical report by [16], in which is was suggested to stabilise this ill-posed process with Tikhonov regularisation and directional filtering.

Regarding our subdivision strategy, many related variable block size image coding algorithms exist, in particular methods based on quadtree decompositions; see e.g. [61] and [62]. Interesting adaptive triangulation ideas can be found in [28], [25], and [13].

In the 3-D setting, there have been no pure PDE-based compression methods so far. However, 3-D diffusion was proposed as a pre- or postprocessing step within transformation-based compression methods such as MPEG4; see e.g. [68] or [14]. A generalisation of image compression ideas with homogeneous diffusion to the compression of surfaces is studied in [6].

The second part of our manuscript extends preliminary results of a conference paper [59]. Substantial differences are, among other things, the comparison of several differential operators within an image compression framework, the investigation of different point selection patterns within our rectangular subdivision, a more refined quantisation, and an improved interpolation swapping step. We will see that these modifications enhance the compression performance. The extension to 3-D data is based upon the work in [55].

2 PDE-Based Interpolation

As explained in the introduction, only the brightness of specific pixels and their positions are stored in our image compression framework. In this section, we explain how the image can be recovered from these sparse data using PDE-based image interpolation.

Let Ω denote the complete domain. For images, we have $\Omega \subset \mathbb{R}^2$, while $\Omega \subset \mathbb{R}^3$ holds for volumetric data. Moreover, let $K \subset \Omega$ be the set of those locations for which the image brightness is known, i.e. for which the brightness is stored in the compressed image. Note that the equations presented in this section are valid for both kinds of data. The set K is called the *interpolation mask*.

The goal behind PDE-based image inpainting is to compute a reconstruction u of the original image $f : \Omega \to \mathbb{R}$ that fulfils the following two properties: First of all, u should be identical to f at those locations for which the brightness is known, i.e.

$$u(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \forall\, \boldsymbol{x} \in K. \tag{1}$$

Secondly, u should own some kind of regularity properties. Typically, u is assumed to be smooth or piecewise smooth outside K. Both properties are fulfilled when using the solution of the following PDE as reconstruction:

$$(1 - c_K)\, Lu - c_K\, (u - f) = 0, \tag{2}$$

with reflecting (i.e. homogeneous Neumann) boundary conditions. Here L denotes some differential operator that guarantees the desired smoothness, and c_K is the characteristic function of K that is 1 at the specified set K, and 0 elsewhere. Since the reconstructed image u does not change at the specified locations K, (2) comes down to the simplified PDE

$$Lu = 0 \tag{3}$$

on $\Omega \setminus K$, with Dirichlet boundary conditions (1) on K, and homogeneous Neumann boundary conditions on the boundary of Ω. This equation can be solved by computing the steady state of the evolution equation

$$\partial_t u = Lu, \tag{4}$$

where t serves as an artificial time parameter. For $t \to \infty$, we thus obtain the reconstructed image.

In the remainder of this section, we give a short introduction on several possible choices for the differential operator L. The simplest differential operator L that can be used for inpainting is the linear diffusion operator [37, 72]:

$$Lu = \Delta u = \operatorname{div}(\boldsymbol{\nabla} u). \tag{5}$$

We will also consider higher order linear operators such as the *biharmonic operator*

$$Lu = -\Delta^2 u, \tag{6}$$

or the *triharmonic operator*

$$Lu = \Delta^3 u. \tag{7}$$

Another interpolation operator that we will evaluate is the *absolute minimal Lipschitz extension (AMLE)* introduced in [4]. This operator which has been axiomatically justified for interpolation tasks by [20] is given by

$$Lu = u_{\eta\eta}, \tag{8}$$

where η denotes the normalised gradient. Since this operator interpolates only in the direction of the gradient, it acts anisotropically.
In [54], it was proposed to prevent smoothing semantically important edges by reducing the diffusion at edges of the evolving image, namely by using the operator

$$Lu = \operatorname{div}(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u), \tag{9}$$

where g is a nonnegative, decreasing function in its argument $|\boldsymbol{\nabla} u|^2$ for which $g(0) = 1$ holds. In our experiments we use the *Charbonnier diffusivity* [24]

$$g(s^2) := \frac{1}{\sqrt{1 + \frac{s^2}{\lambda^2}}}, \tag{10}$$

where $\lambda > 0$ is a contrast parameter. Thus, we call the corresponding interpolation operator *Charbonnier diffusion*. With this diffusivity we have observed better interpolation results than for the more rapidly decaying diffusivities of [54]. For our interpolation experiments it will be instructive to study also a regularised variant of the Charbonnier operator. It is given by

$$Lu = \operatorname{div}(g(|\boldsymbol{\nabla} u_\sigma|^2)\, \boldsymbol{\nabla} u), \tag{11}$$

where $u_\sigma := K_\sigma * u$ is the smoothed image obtained by convolving u with a Gaussian K_σ with standard deviation σ. We call (11) the *regularised Charbonnier operator*. In the context of diffusion filtering, this regularisation has been introduced by [21] in order to make the Perona-Malik filter well-posed and more stable under noise.
Last but not least, let us also study a direction-dependent (and therefore anisotropic) variant of a nonlinear diffusion operator which reduces smoothing across edges

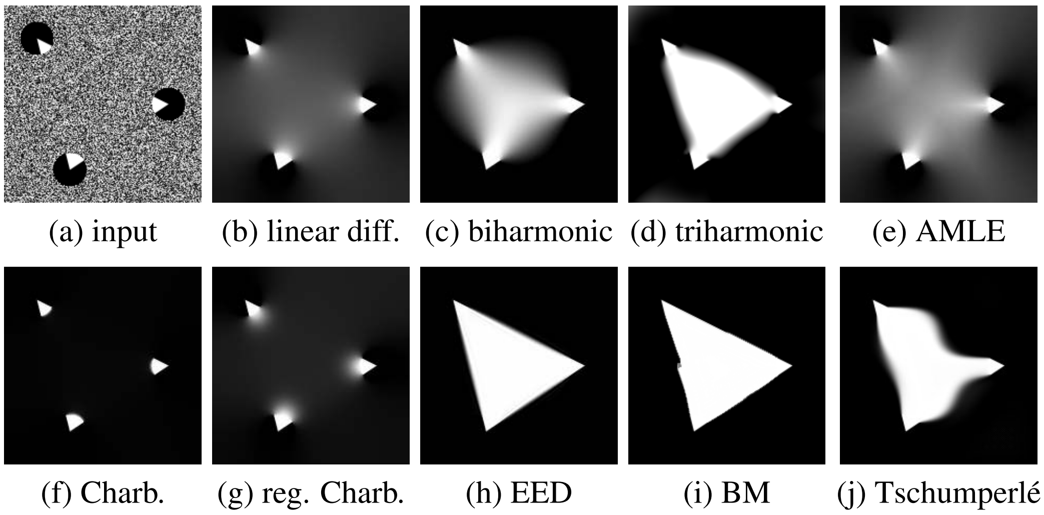
while still permitting diffusion along them. It has been introduced for diffusion filtering in [69] and is named *edge-enhancing (anisotropic) diffusion (EED)*. For image compression it was first used in [32]. The differential operator of EED replaces the scalar-valued diffusivity $g(|\boldsymbol{\nabla} u_\sigma|^2)$ in (11) by a matrix-valued diffusion tensor $\boldsymbol{g}(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top)$:

$$Lu = \operatorname{div}(\boldsymbol{g}(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top)\,\boldsymbol{\nabla} u). \tag{12}$$

Thereby, we extend the scalar-valued function $g(x)$ to a matrix-valued function $\boldsymbol{g}(\boldsymbol{A})$ by applying g only to the eigenvalues of $\boldsymbol{A}$ and leaving the eigenvectors unchanged. Again, we use the Charbonnier diffusivity given in (10). Since EED is designed such that it smoothes along edges but very little across them, this diffusion process respects not only the location but also the direction of sharp edges. Moreover, the Gaussian smoothing within $u_\sigma = K_\sigma * u$ allows to propagate this anisotropy also to the vicinity of the specified data. We will see that these are important properties for interpolation.
Furthermore, we will evaluate two PDE-based inpainting algorithms introduced by Tschumperlé [65], and by Bornemann and März [11], respectively. The approach by Tschumperlé is a tensor-driven PDE which takes the curvature of specific integral curves into account, while the second is an extension of Telea's single-pass algorithm [64] that uses an exponentially confining weight function. Since a detailed explanation of these algorithms is beyond the scope of this paper, we refer to [65], [11], and [64] for more details.

3 Comparison of Differential Operators

To evaluate the different PDE-based interpolation operators from Section 2, we first compare them in several illustrative scenarios, starting with the example shown in Fig. 1. From three black discs with white wedges given as initial data, the remainder of the image is to be reconstructed. In this example, which is similar to the well-known Kanizsa triangle, a human observer sees a triangle.
Even though only points within three small disks are given, EED yields an almost perfect reconstruction of the entire triangle. This favourable performance can be understood as follows: The anisotropy allows EED to create sharp edges, while the Gaussian presmoothing within the diffusion tensor propagates a directional preference also to areas outside the specified disks. Hence, EED features two important qualities: *anisotropy* and *semilocality* (in the sense that it is not purely local, but also involves the neighbourhood).
The method by Bornemann and März also creates an almost perfect triangle, as this approach is good at connecting level lines. This explains why it works very


(a) input (b) linear diff. (c) biharmonic (d) triharmonic (e) AMLE

(f) Charb. (g) reg. Charb. (h) EED (i) BM (j) Tschumperlé

Figure 1: **(a) First Image**: Initial image, 189×189 pixels. The brightness values inside the three disks are specified, while the remainder was initialised with uniform noise. **(b)-(h) Remaining images**: Reconstruction results with linear diffusion interpolation (see (5)), biharmonic interpolation (see (6)), triharmonic interpolation (see (7)), AMLE (see (8)), Charbonnier interpolation (see (9), $\lambda = 1$), regularised Charbonnier interpolation (see (11), $\lambda = 0.1$, $\sigma = 8$), EED interpolation (see (12), $\lambda = 0.01$, $\sigma = 4$), the method from [11] ($\varepsilon = 5, \kappa = 100, \sigma = 4, \rho = 8$), and the method from [65] (30 global and local iterations, $dt = 17.86, \alpha = 2.81, \sigma = 0.27$).

well for this example. However, one has to choose the parameters very carefully, as even slight variations significantly alter the results. The method by Tschumperlé, which was created with the G'MIC plug-in for the GIMP, yields rather poor results. Moreover, the algorithm is quite slow and did not converge even after 5000 iterations. This is more than 20 times the number of iterations used in the original paper. Thus, we do not further evaluate this method in our compression framework.
The unsatisfactory results of the other interpolation operators show that the success of EED lies in the *combination* of the two properties explained above. Apart from AMLE, all other operators are isotropic. Unfortunately, AMLE prefers a direction perpendicular to isophotes. The fact that biharmonic and triharmonic interpolation create some blurry triangle-like structure is caused by their higher degree of smoothness, not by any kind of anisotropy. Gaussian smoothing within regularised Charbonnier interpolation gives a better propagation of structures than unregularised Charbonnier interpolation. However, regularised Charbonnier interpolation cannot restore the triangle due to its lack of anisotropy.

It is also interesting to observe that unregularised Charbonnier interpolation does not propagate any white area of the disks. Since it approximates total variation (TV) interpolation and most of the data at the boundary of the disks are black, the total variation is minimised by a black interpolant. This shows that isotropic TV ideas can be problematic for compression tasks, since TV may ignore a substantial part of the data.

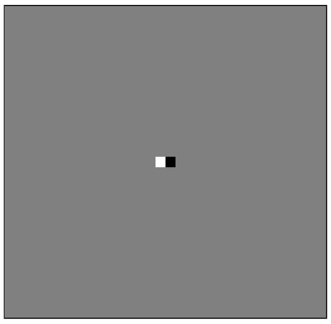
It should be noted that using TV in an anisotropic way can be highly successful, though: EED combines linear diffusion interpolation along edges with (an approximation of) TV interpolation across edges. A vanishing diffusion across pronounced edges creates a segmentation of the interpolation problem into subproblems where the specified grey values are similar if they belong to the same segment. Inside each segment, essentially isotropic linear diffusion interpolation is performed. However, since the grey values are similar, the main problem of isotropic linear diffusion interpolation, namely logarithmic singularities for fluctuating data, is not present.

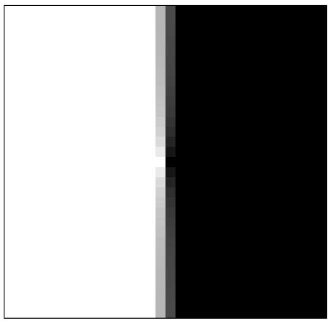
After we have investigated the differences between EED and isotropic second order differential operators for interpolation, let us now study the difference to higher order differential operators in more detail. This is done in Fig. 2, where we interpolate a "dipole" image in which only two adjacent pixels of black and white colour are specified. We see that both EED and biharmonic interpolation are capable of segmenting the image plane into two halfplanes. However, EED creates sharper boundaries due to its pronounced anisotropic behaviour and the property of TV interpolation to admit jumps across the edge. Biharmonic interpolation, on the other hand, aims at creating a smooth interpolant, also across edges. Moreover, this smoothness is at the expense of violating a maximum-minimum principle, as can be clearly seen from the over- and undershoots at both sides of the dipole.

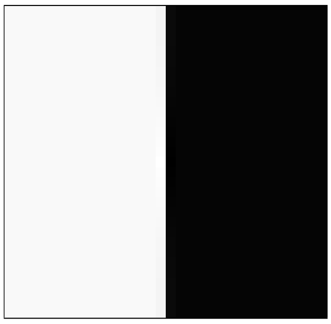
Our dipole experiment has motivated us to perform another minimalist experiment: Fig. 3 sheds light on the capabilities of EED to encode shapes by means of a sparse set of pixels. We see that four dipoles are sufficient for EED interpolation to reconstruct the shape of a disk. This illustrates that EED also has a tendency to favour shapes of low curvature. The Gaussian smoothing within the diffusion tensor encourages such rounding effects.

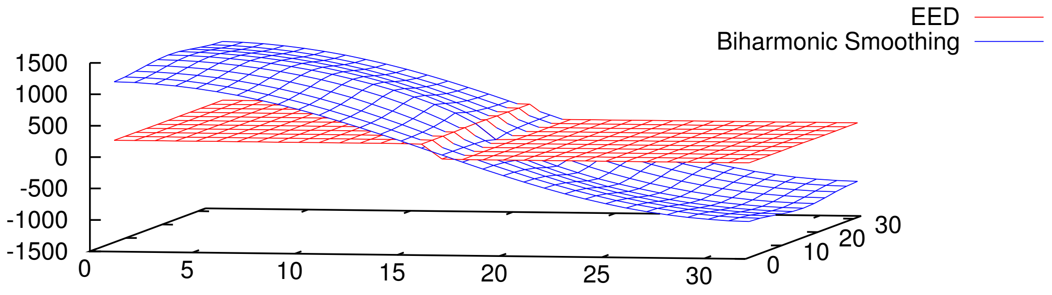
4 Our Image Codec

After we have discussed the principles of PDE-based interpolation and evaluated several differential operators in the previous sections, let us now introduce our image compression framework that decides which pixels are kept. Moreover, we will detail on how these pixels can be stored efficiently.


(a) input image


(b) biharmonic


(c) EED



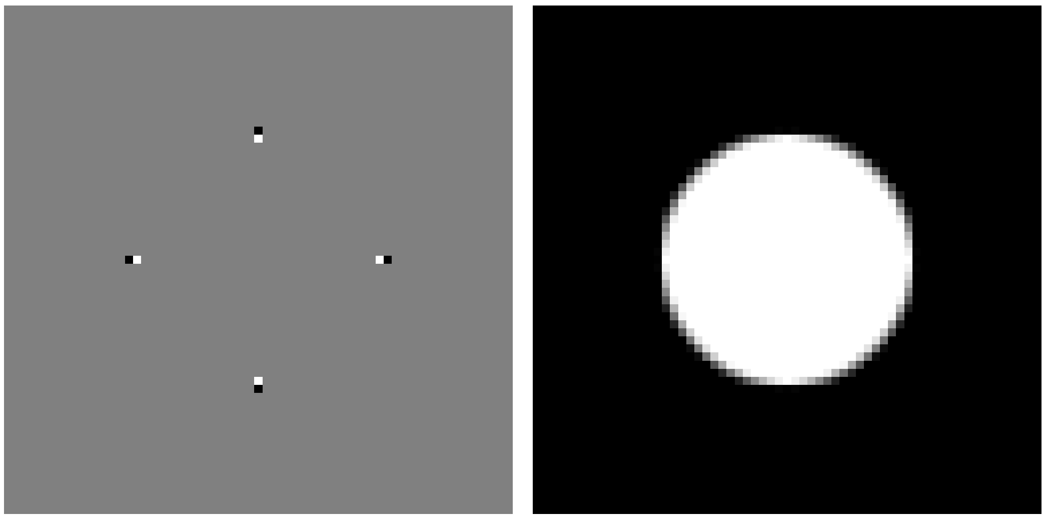
(d) results plotted as height field

Figure 2: **(a) Top left**: Input image, 32×31 pixels. Only the black and the white pixels in the middle of the image are specified. **(b) Top middle**: Interpolation result with the biharmonic operator. **(c) Top right**: Interpolation result with EED. **(d) Bottom**: Results of the two interpolation approaches plotted as height field. Biharmonic interpolation produces large over- and undershoots, while EED stays within the specified grey value range $[0, 255]$.

4.1 Selecting and Encoding Pixel Locations

Deciding which pixels of a given image should be keep is a difficult task, as there is little theoretical knowledge about how these pixels should be chosen. One notable exception is homogeneous diffusion, for which analytical answers to this question have been derived using the continuous theory of shape optimisation [9]. However, even in this case there are still remaining degrees of freedom depending on the model assumptions and the discrete implementation of continuous results [47]. Moreover, optimal interpolation in an approximation theoretic sense may not be optimal for compression if there are suboptimal pixel masks that can be encoded more efficiently.

One obvious possibility would be a simple subsampling approach, where all pixels on a regular subgrid are kept. While this avoids additional costs for storing the pixel locations, such a nonadaptive strategy usually does not give a satisfactory reconstruction quality: In flat image areas, too many points are kept while they



(a) input image (b) EED interpolation

Figure 3: **(a) Left:** Original image of size 63×63 pixels, where 4 dipoles are specified. **(b) Right:** After EED interpolation $(\lambda = 0.01, \sigma = 1)$.

may be too sparse in more interesting areas, e.g. near edges.
Another natural idea is to try all possible subsets (with a specified number of pixels) of the set of all pixels, and to choose the subset that yields the best results. Since this is a finite combinatorial problem, we are guaranteed that an optimal solution exists. However, for all but the smallest images, this is infeasible due to the large amount of possible subsets: Selecting e.g. 10 % of the pixels of a 256×256 image leaves

$$\binom{65536}{6554} \approx 3.8 \cdot 10^{9250} \tag{13}$$

options. Moreover, saving the positions of the points is quite expensive in this case, since there is no regular pattern behind the position of optimal points.
As a remedy, we now propose a method that restricts the search space by use of an adaptive rectangular grid that allows to store the pixel positions in an inexpensive way in a tree structure. We first approximate the image with a few points at fixed positions, e.g. the four corners of the image, and reconstruct the image using one of the inpainting schemes described in Section 2. If the reconstruction is not accurate enough, the image is split into two subimages in the middle of the x or y direction, whichever is larger, and the two subimages are saved recursively. This splitting is illustrated in Fig. 4, and a final mask in Fig. 5.
In order to decide if the reconstruction quality is sufficient, i.e. whether a splitting is performed, we compute the mean square error (MSE) between original and

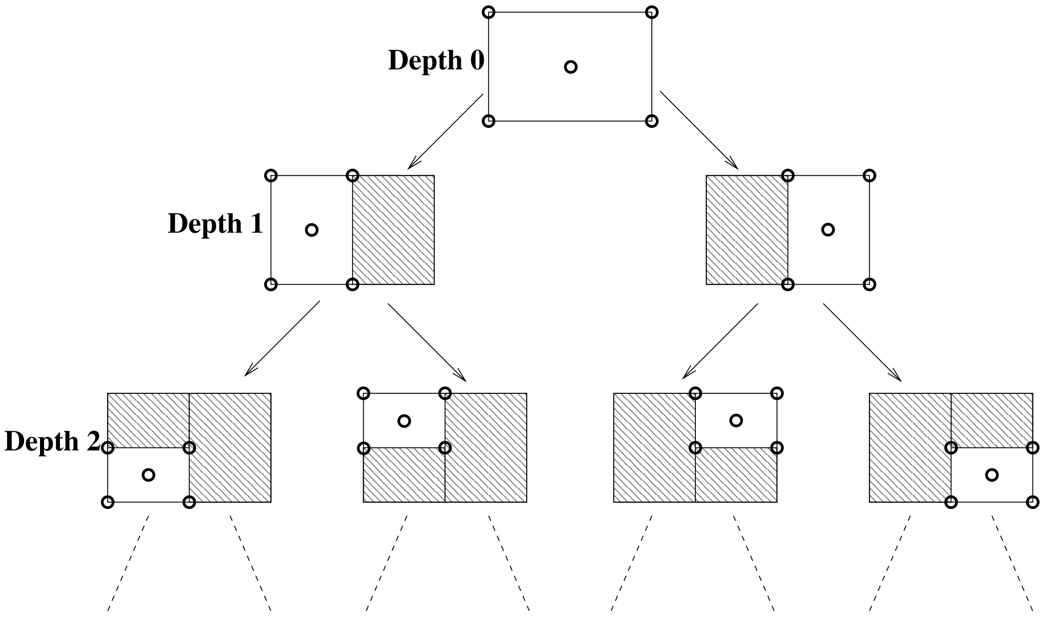


Figure 4: Illustration of the rectangular subdivision scheme. The white parts are the (sub)images being processed, while the circles show one example for the points saved in this subimage. For this example, Pattern F from Fig. 6 was used.

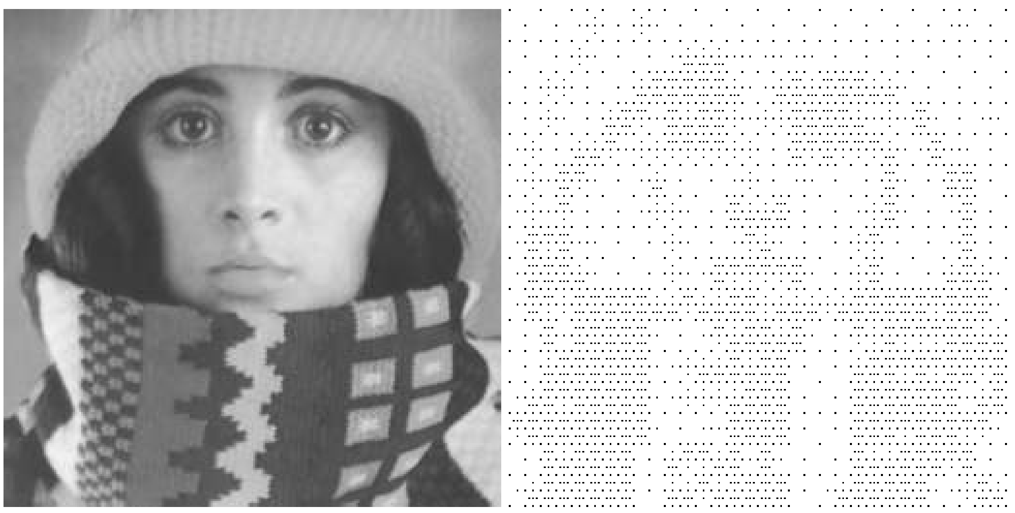


Figure 5: **Left**: Input image “trui” used for several experiments. **Right**: Example of an interpolation mask used for this image.

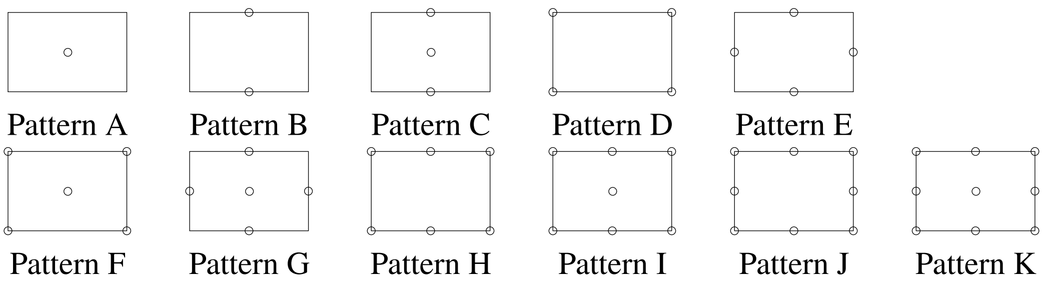


Figure 6: The eleven pixel selection patterns we evaluated in our image compression framework. In Pattern B, the two pixels shown are always along the longer side of the rectangle. The same is true for the same two points in Pattern C, H, and I.

reconstructed image part. The MSE is then compared with a threshold. If it exceeds this threshold, we subdivide. The threshold T is given by $T := a\ell^d$, where d is the recursion depth and where a and ℓ are free parameters (see [32]). To reduce the size of the tree in the compressed file (see below), it often makes sense to use these thresholds only for a limited recursion depth, i.e. all images are split up to a certain level, and no split is done once a maximal recursion depth is reached.
In [59], we assumed that the complete subimage boundary is known when deciding whether to subdivide a part of the image. However, since only a few key points are stored, the actual reconstruction is not based on the whole boundary. Thus, in the present article, we use only those points for the compression step that are candidates for being saved.
The location of the stored pixels is uniquely determined by the splitting decisions for the subimages, assuming the saved points in each rectangle are known. Thus, instead of storing the position of each saved pixel individually, it is sufficient to save the binary tree containing the splitting decisions. We store this tree by first saving its minimal and maximal depth, followed by one bit for each tree edge between these two depths. Note that the amount of space necessary to save the minimal and maximal tree depths depend logarithmically on the image dimensions.
Let us now discuss how to choose the mask points in each subimage. A straightforward idea would be to select the two endpoints on the line used to split the image, as well as the point in the middle of this line, as done in [59]. Here we investigate different alternatives for suitable point sets to be stored.
For our experiment we compare the quality obtained with the different point selection patterns. Fig. 6 shows the different point patterns we evaluate, and Table 1 the best MSE obtained with each pattern for three different images. While each novel line is approximated by three points in [59], which corresponds to Pattern C, we observe that pattern F yields better results. This is due to two reasons: First

Table 1: Number of points in each of the patterns shown in Fig. 6, and mean square error (MSE) for each pattern when saving different images with a compression rate of 20 : 1. The best result in each line is highlighted. No entropy coding was performed for these experiments. The original images are shown in Fig. 7 and 10, respectively.

Pattern	A	B	C	D	E	F	G	H	I	J	K
No. of points per subimage	1	2	3	4	4	5	5	6	7	8	9
MSE for image “trui”	47.75	46.49	43.98	38.08	47.53	**35.78**	46.94	37.60	39.68	44.09	41.86
MSE for image “walter”	23.61	25.13	22.51	20.64	23.27	**19.95**	24.58	20.27	22.30	22.44	20.90
MSE for image “peppers”	56.50	53.29	54.50	**44.13**	56.30	45.88	59.05	46.22	50.83	52.70	50.36

Table 2: Effect of entropy coding of pixel data from the image “trui”, with varying splitting thresholds in the subdivision process. Shown are the size of the compressed pixel data in bytes without entropy coding (raw), with Huffman coding (HC), with Huffman coding using canonical codes (HCc), with arithmetic coding with static (ACs) or adaptive (ACa) model, PAQ, gzip, bzip2, Lempel-Ziv-Welch Coding (LZW), and range coding (RC). The best results are highlighted.

Raw	HC	HCc	ACs	ACa	PAQ	gzip	bzip2	LZW	RC
7587	3596	3773	3664	3515	**2745**	3439	3259	3545	3985
3789	1850	2027	1906	1788	**1536**	1851	1797	1977	2258
1897	973	1150	1012	923	**855**	975	984	1072	1394
950	530	707	548	487	**485**	512	533	600	958
469	302	479	297	**263**	283	284	305	325	734
199	175	352	145	**139**	154	151	169	164	610

of all, the number of points in the pattern serves as a tradeoff between accurate point localisation and overhead necessary to store the binary tree: In patterns with a low amount of points, more rectangles are necessary to store a certain amount of point positions. Thus, a large tree is needed to encode the positions of a certain number of points, resulting in too much overhead to obtain a good approximation. In patterns with many points, there is little overhead (compared to the number of pixels saved), but the point locations are more restricted, again giving suboptimal results. This explains the ordering of all patterns except for Patterns E, G, and J. Here, the problem is that the points are not distributed well over the complete domain. As Patterns D and F have the right number of well distributed points, they yield the best results. We should mention that Pattern D yields good results especially for high compression ratios, as the tradeoff described above slightly favours using less points in case of high compression ratios.

4.2 Quantising and Encoding the Brightness Data

Up to now we have only stored the locations of the points that are kept in a lossless way. In the following we explain a lossy strategy for storing the corresponding brightness values. They are obtained by scanning the interpolation mask from top left to bottom right.

For each point in the mask, the corresponding brightness value is first requantised. Reducing the quantisation levels from the original 256 levels of bytewise coding to a smaller amount allows to save many bytes. These savings can be invested in additional pixels that are stored. There is of course a tradeoff, since a coarser quantisation also deteriorates the approximation quality in the individual pixels. For simplicity we use an equidistant quantisation. In our current implementation this quantisation step may use any integer number of quantisation levels, while this number was restricted to a power of 2 in [59]. Renouncing powers of 2 is no problem, if we apply some suitable entropy encoding of the grey levels. Let us discuss this next.

The quantised pixel values are encoded in a lossless way using a general purpose entropy coder. We tested several different entropy coders including Huffman coding [35], arithmetic coding with static or adaptive model [57], Lempel-Ziv-Welch coding [74], gzip (version 1.3.5), bzip2 (version 1.0.3), and a slightly modified version of PAQ (version paq8o8z-feb28, [45]). Except for gzip and bzip2, which are standard tools, and PAQ, the source code of which is available at [46], we used the implementations from [27] here.

A comparison of the performance of the different entropy coders can be found in Table 2. In the experiments shown in Section 5 and 6, we always use the best entropy coder for a given problem (it is stored in the header which coder must be used to decompress the data). This was either PAQ, or, for very small files,

arithmetic coding with an adaptive model.
Allowing an MSE of 54.46 for the image "trui", we obtain a compression ratio of 40.02 : 1 using arithmetic coding with an adaptive model as entropy coder for the brightness values. Using PAQ, the compression ratio even rises to 47.46 : 1. In the following sections, we use this as ongoing example to demonstrate the effect of each of our optimisation steps.

4.3 Diffusion Parameter Optimisation

The differential operators from nonlinear diffusion methods involve a contrast parameter λ. It helps to distinguish between high contrast locations where the diffusion is reduced and low contrast regions where one is interested in approximating homogeneous diffusion. While [33] used a constant λ, we noticed that the results can be improved by adapting λ to the image and the desired compression ratio. Therefore, we search for the contrast parameter which yields the best reconstruction result and store it in the header of the compressed image.
The contrast parameter is always positive by construction, and the optimal values have a limited range. Thus, we can quantise this range into 256 values in a linear way, and use one byte to store it. Note that the range differs depending on whether Charbonnier interpolation, regularised Charbonnier interpolation or EED interpolation is considered. For example, the interval $[0,1]$ is a useful range when applying EED to images with greyvalue range $[0,255]$. In this case we are fairly close to TV interpolation across edges. Here, we tested each of the 256 possible values to find the best λ, but using a golden section search yields only slightly worse results.
In the image "trui", optimising the contrast parameter improves the MSE from 54.46 to 53.35. In this example, this is only an incremental gain. For other images, however, this step may lead to larger improvements.
Since we optimise the contrast parameter λ, a natural question would be if it makes sense to optimise also the regularisation parameter σ. It determines the amount of Gaussian smoothing within the diffusivity of regularised Charbonnier interpolation and within the diffusion tensor of EED interpolation. We have also tried this and experienced that even variations by a factor 10 have only a very limited impact on the compression quality. It seems that the pure presence of some regularisation is more important than its precise amount. Thus, in our compression experiments with pixel size 1 we have decided to fix σ to 0.8.

4.4 Brightness Optimisation

It is possible to significantly improve the reconstructed image by adapting the brightness of the stored pixels. Obviously, this step introduces an additional error

at these pixels. However, the resulting image has a much higher overall reconstruction quality.
We currently use a straightforward approach: Our programme looks at each pixel in the mask in a random order, and checks if increasing or decreasing its brightness to the next quantisation level reduces the reconstruction error. If this is the case, the new brightness value is stored. The algorithm stops if no single point can be optimised any more. More advanced optimisation schemes are also possible, as shown in [47].
Modifying the brightness values of the selected pixel set K can lead to remarkable improvements: Optimisation in a single pass reduces the MSE of our example image “trui” from 53.35 to 36.48. With multiple optimisations, the MSE improves further to 31.08.

4.5 Interpolation Swapping

If we use the above mentioned steps, an interesting phenomenon can be observed: It can happen that the reconstruction quality within the “interpolation” set K is worse than in the inpainting domain $\Omega \setminus K$, especially for high compression ratios. There are three reasons for this behaviour:

(i) A coarser quantisation of the stored brightness values creates errors.

(ii) Higher errors are accepted in the interpolation set K if this is beneficial for the approximation quality within the inpainting domain $\Omega \setminus K$.

(iii) In the domain $\Omega \setminus K$, data are inpainted by PDEs that average information from K. Even if the stored brightness values in K are erroneous due to requantisation and brightness optimisation steps, a blend of them averages the errors and may lead to improved results in the inpainting domain $\Omega \setminus K$. This resembles variational optic flow estimation where the results at locations with a large data term are worse than in areas where the smoothness term with its inpainting effect dominates [17].

Moreover, for isotropic linear and nonlinear diffusion interpolation it can be observed that inpainting results may become singular when approaching an isolated pixel from the set K. In the linear case this can be explained with the well-known logarithmic singularity of the Green’s function of the 2-D Laplacian.
Interestingly there is a relatively simple remedy for all these problems: In [6] it is proposed to perform an additional inpainting step in which the role of known and unknown pixels is swapped. That is, after reconstructing the image with inpainting, one regards the reconstructed points in $\Omega \setminus K$ as known and the specified points in K as unknown. Then one inpaints the data in K using the information

from $\Omega \setminus K$. This leads to an overall solution that is smoother and propagates the high quality solution from $\Omega \setminus K$ to the more erroneous data in K. While this strategy may appear ad hoc at first glance, it can be justified as a numerically consistent approximation of the inpainting PDE in the sense of so-called Hopscotch schemes [34].

Here we propose a modification that extends the interpolation swapping of [6]: Instead of only recomputing the known points, we also allow the possibility to recompute more points. More precisely, we reconstruct all points within a certain radius around the saved mask points. The radius for which the best reconstruction is achieved is stored in the file header.

Since the best brightness values and the optimal contrast parameter λ can change during these optimisations, brightness optimisation, diffusivity optimisation, and finding an optimal interpolation swapping are interleaved.

The radius that yields the best result is stored in the header of the compressed file and used in the reconstruction step. For our test image "trui" this step reduces the MSE from 31.08 to 28.29.

4.6 Colour Images

Very little is necessary to extend our algorithm to colour images. Here, we simply save the R, G, and B value of each pixel indicated by the inpainting mask in an interleaved way. The file containing all colour channels is then compressed together using the chosen entropy coder. An additional bit in the file header indicates whether a grey-scale or colour image has been stored. In the decompression step, we reconstruct all three channels simultaneously using vector-valued inpainting operators [71]. Note that this yields different results than inpainting each channel separately when using a nonlinear inpainting process, since the argument of the diffusion tensor depends on all channels.

4.7 Decoding Algorithm

In the decoding step our codec creates the inpainting mask from the splitting information and the image size. The pixel values at these positions are obtained by decoding the pixel data with the entropy coder and inserting them at the appropriate positions. Then the remainder of the image is inpainted using the same inpainting method as in the compression step. Finally, interpolation swapping is performed.

4.8 Numerical Implementations

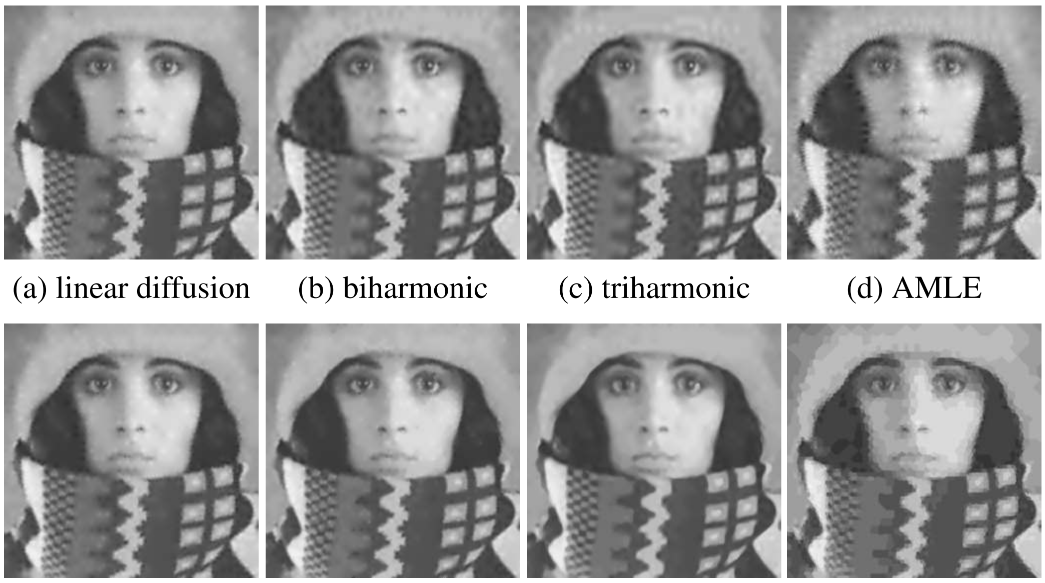
The PDEs we consider have been implemented using a variety of numerical schemes. Most of them are based on finite difference discretisations of the evolution equation given in (4). To allow for large time steps for the diffusion schemes, (semi-)implicit time discretisations have been implemented which are solved using e.g. SOR or conjugate gradients. AMLE interpolation has been implemented with an explicit scheme, and for the biharmonic and triharmonic interpolation methods also pseudospectral methods are used. The actual CPU times depend strongly on the choice of the differential operator, the numerical scheme, the optimisation strategy, and the hardware. Since the present paper focusses on modelling and quality issues, a detailed numerical evaluation will be presented in a forthcoming paper that is entirely devoted to these aspects. It should be mentioned that it was already demonstrated in [41] that one can achieve real-time performance for diffusion-based codecs on a Playstation 3 with images of size 320×240 pixels.

4.9 File Format

The file format of our image compression framework has the following structure:

- image size (between 10 and 34 bits)
- type of inpainting differential operator (3 bits)
- colour flag (1 bit)
- type of entropy coder (4 bits)
- contrast parameter (1 byte)
- type of interpolation swapping (4 bits)
- number of quantisation levels (1 byte)
- minimal / maximal depth of tree (at most 2 bytes)
- splitting information of the tree (variable size)
- compressed brightness values (variable size)

Ignoring the splitting information of the tree, our header thus occupies less than 10 bytes.


(a) linear diffusion (b) biharmonic (c) triharmonic (d) AMLE

(e) Charbonnier (f) reg. Charbonnier (g) EED (h) BM

Figure 7: Reconstruction from optimised point masks with linear diffusion interpolation (see Equation (5)), biharmonic interpolation (see Equation (6)), triharmonic interpolation (see Equation (7)), AMLE (see Equation (8)), Charbonnier interpolation (see Equation (9)), regularised Charbonnier interpolation (see Equation (11)), EED interpolation (see Equation (12)), and the method by Bornemann and März (BM). Each point mask is optimised such that the smallest MSE for a compression ratio of 45 : 1 is obtained. A quantitative evaluation of these images is given in Table 3.

5 Evaluation of our Compression Framework

In this section, we first compare the performance of different PDE-based inpainting operators within our image compression framework. Afterwards, a comparison against various image compression algorithms is performed.

5.1 Comparison of the Differential Operators in Our Compression Framework

To further evaluate the different PDE-based interpolation operators from Section 2, we used them to compress images with our compression framework introduced in the last section. Here, the image “trui” with a compression rate around 45 : 1 is used. All parameters have been chosen in such a way that optimal point masks, quantisation levels, and tree depths for each method are obtained. The results are

Table 3: Interpolation error obtained in our compression framework with different interpolation operators and a compression rate of 45 : 1. The corresponding images are shown in Fig. 7. **First line**: MSEs obtained without optimising grey values, diffusion parameters, and interpolation swapping. **Second line**: MSEs obtained with optimisations.**Third line**: Number of quantisation levels. **Fourth line**: Number of points saved. **Fifth line**: Depth range of the adaptive tree structure.

	Linear	Biharmonic	Triharmonic	AMLE	Charb.	Regul. Charb.	EED	BM
MSE	93.00	59.89	75.77	148.04	93.02	70.42	53.41	99.32
Optimised MSE	35.40	39.79	42.83	64.94	35.25	29.53	28.29	69.45
Quantisation levels	11	20	17	11	11	11	24	17
Points saved	3502	2858	3021	3469	3510	3516	2750	2991
Depth range of tree	9–13	10–13	10–13	8–13	9–13	9–13	9–12	9-13

Table 4: Compression results for the images "trui", "walter", and a grey-valued subimage of "peppers" with JPEG, the method from Galić et al. [33], our preliminary conference paper [59], and the proposed algorithm (R-EED).

	JPEG		**Galić et al.** [33]		**JPEG 2000**		**Preliminary results** [59]		**R-EED**	
	Ratio	MSE	Ratio	MSE	Ratio	MSE	Ratio	MSE	Ratio	MSE
trui	42.13 : 1	71.16	43.67 : 1	53.33	43.44 : 1	45.99	44.11 : 1	31.00	46.53 : 1	28.29
walter	45.11 : 1	39.67	45.58 : 1	30.84	45.78 : 1	27.75	45.40 : 1	20.13	49.36 : 1	16.43
peppers (BW)	42.03 : 1	70.47	43.93 : 1	57.27	44.10 : 1	50.95	42.96 : 1	42.61	46.26 : 1	37.17

shown in Fig. 7 and Table 3.
We see that the best results are obtained with EED. This is true with respect to the MSE (see first rows in Table 3) as well as visually: The method by Bornemann and März creates too many edges, i.e. it oversegments the image. The results from biharmonic or triharmonic interpolation show visible fluctuations due to over- and undershoots at edges. This is characteristic for higher-order differential operators that violate a maximum-minimum principle. The other second-order operators (linear diffusion interpolation, AMLE, standard or regularised Charbonnier interpolation) suffer from the fact that the information in the selected pixels seems to be too sparse to create regular edge contours that do not appear blurry.
The implicit segmentation performed by EED (see Section 3) explains the absence of point-like singularities in the EED interpolation of Fig. 7(h), which can be found by carefully examining the results of the isotropic second-order operators in Fig. 7(b),(f),(g). The singular behaviour of EED interpolation is restricted to singularities *across* pronounced edges, which is desirable in image processing.
Analysing Table 3 indicates that EED needs a smaller number of points to create good results. Moreover, a precise localisation (given by a high depth range in the tree) seems to be less important than for the other methods. For a specified compression ratio, these savings can be invested in a larger number of quantisation levels that allow a better approximation quality.

5.2 Comparison to Other Compression Methods

Next we compare our compression algorithm with the one proposed by Galić et al. [33] and with the image compression standards JPEG and JPEG 2000. Fig. 8 shows the results for the image "trui". We used the tool "convert" (version ImageMagick 6.6.0-4 2012-05-02 Q16) to create the JPEG and JPEG 2000 images shown here.
Although "convert" uses optimised entropy coding parameters when saving JPEG files, we observe that JPEG is clearly outperformed by the other approaches. Moreover, block artifacts become visible. They are characteristic for JPEG results at high compression rates, since JPEG subdivides the image into 8×8 pixel patches and quantises the coefficients of the discrete cosine transform within each patch.
Interestingly the more recent and more advanced JPEG 2000 standard also produces a noticeably larger mean square error than our R-EED approach: For the shown compression ratios around $56:1$ it is 106% worse. As visible in the graph shown in Fig. 8, the advantages of R-EED over JPEG and JPEG 2000 increases with the compression rate.
Although it was claimed in [33] that this approach was not able to beat the quality of JPEG 2000 for classical test images, recent experiments demonstrate that this is

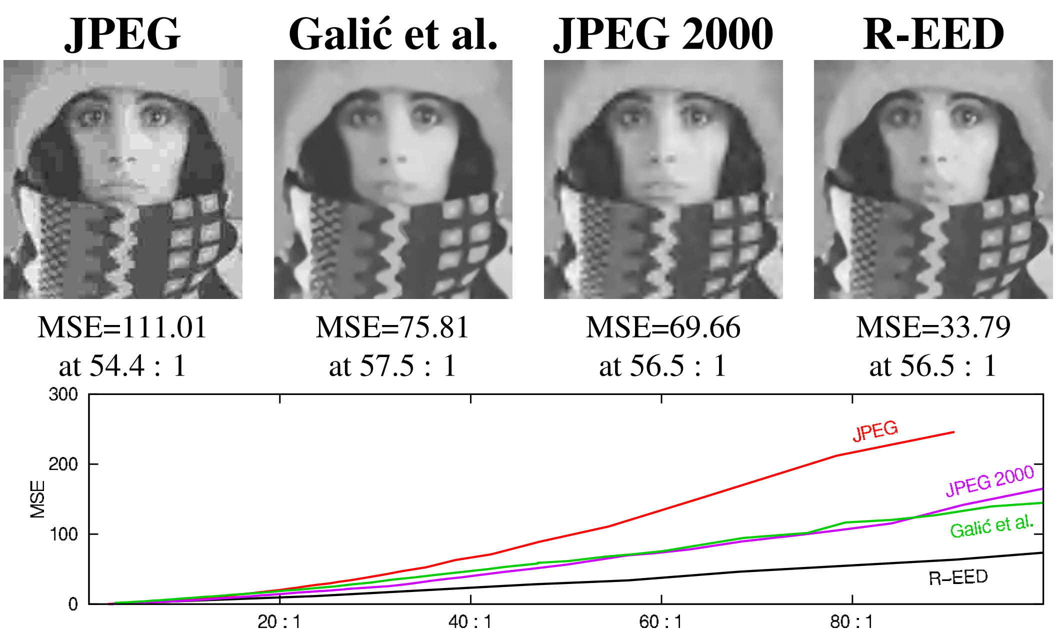
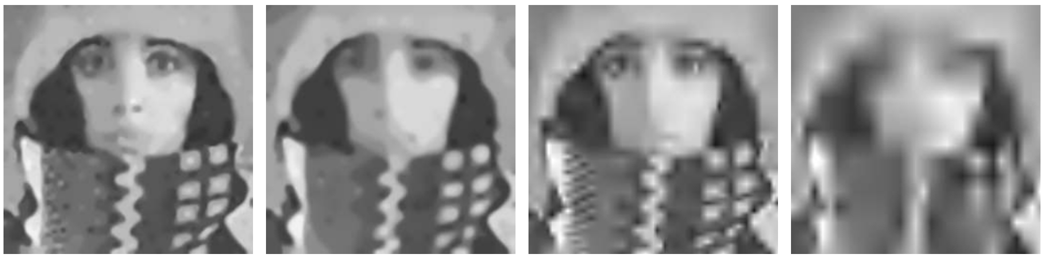


Figure 8: **First row**: Images obtained with JPEG, the method of Galić et al. [33], JPEG 2000 and with the proposed method (R-EED) with a compression rate close to 56 : 1. **Bottom row**: Plot showing the MSEs of different compression algorithms and compression rates.

possible when considering very high compression ratios. However, the results of our algorithm outperform this method. For the shown compression ratio, the MSE is 124% worse. Since both algorithms are based on EED as the differential operator of choice, this clearly demonstrates the importance of the additional features of our algorithm, such as rectangular instead of triangular subdivision, diffusion parameter optimisation, more advanced entropy encoders, improved brightness optimisation, and interpolation swapping.
While the performance of PDE-based inpainting methods is known to deteriorate in highly textured regions, it is remarkable that R-EED still gives a better visual impression than JPEG 2000 in the textured regions of the scarf. Obviously, the distinctive advantages of EED near edges allow to spend more points in order to reconstruct highly textured regions more faithfully.
The previous experiment suggests that R-EED is particularly useful for obtaining high compression rates with good quality. This is studied in more detail in Fig. 9 that displays results with compression ratios of up to 211 : 1. Although the visual quality deteriorates, one can still recognise the essential image contents. Higher compression rates are also possible with R-EED.
Finally we evaluate our image compression algorithm with two additional stan-



(a) R-EED 113 : 1 (MSE=87.6) (b) R-EED 211 : 1 (MSE=157.7) (c) JPEG 2000 113 : 1 (MSE=193.8) (d) JPEG 2000 202 : 1 (MSE=547.5)

Figure 9: Results with very strong compression with R-EED (top row) and JPEG 2000 (bottom row).

dard test images: an image of Walter Cronkite ("walter") and a subimage of the colour image "peppers". Both images are available from the SIPI webpage `http://sipi.usc.edu/database` of the University of Southern California. In all cases, the proposed R-EED compression algorithm outperforms JPEG and JPEG 2000 for medium to high compression rates. Details are given in Table 4 and Fig. 10. In this table, one can also see that the MSE difference between our novel results and our preliminary results from [59] can even exceed 20 % for some images.

6 Extensions

In the third and final part of this paper, we introduce two extensions suitable for shape coding as well as for coding of 3-D data.

6.1 Shape Coding

Due to our rectangular subdivision scheme, the stored points are always (more or less) spread along the whole image domain. However, placing a point exactly to a certain position is very costly. This is no big problem for natural images. When encoding shape data, most points are far away from the contour of the shape, though. This results in a high overhead. Thus, we propose an extension suitable for shape coding that can position the stored information arbitrarily.
First of all, we note that a single point is insufficient to encode a meaningful contour information, and that even dipoles do not provide enough flexibility to encode arbitrary lines. In general, they can only be considered to encode vertical or horizontal directions. In order to encode lines with arbitrary directions, we thus

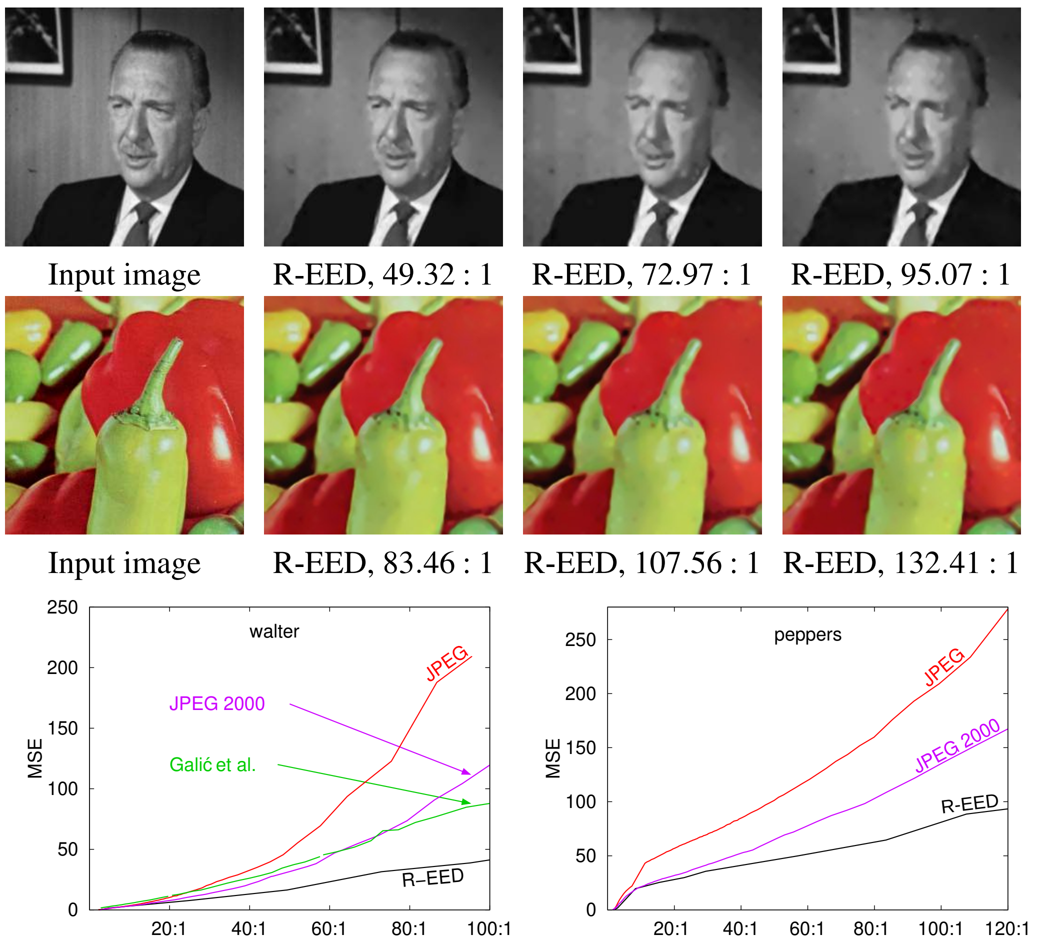


Figure 10: Compression results of the images "walter" (256 × 256 pixels) and of a 256 × 256 subimage of the image "peppers". The first two rows show the input images and images created by the proposed algorithm with different compression ratios. The last line shows two plots that compare the quality obtained with JPEG, JPEG 2000, our algorithm (R-EED). For the first image, the results of Galić et al. are additionally given.

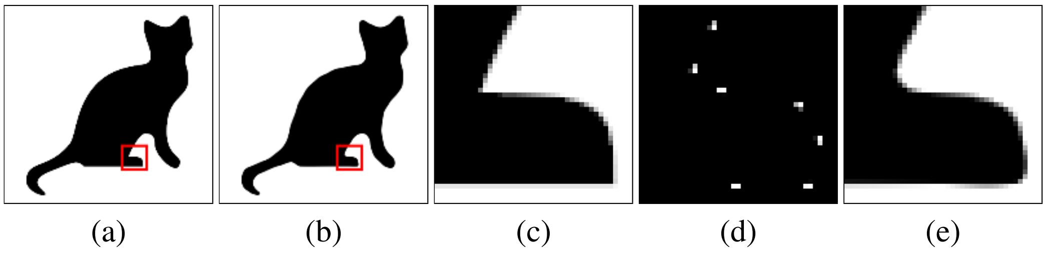


Figure 11: Illustration of the EED shape coding capabilities with 4% of all quadrupoles (67 of 1672); the red rectangles in (a) and (b) mark a difficult area for our algorithm, which is zoomed in (c), (d), and (e). **From left to right**: **(a)** Original shape (400 $\times$ 380 pixels). **(b)** Interpolation result with EED, MSE: 77.79. After thresholding, the binary images only differ in 464 pixels, i.e. in approximately 0.3% of the pixels. **(c)** Zoom into marked area of (a). **(d)** Corresponding selected quadrupoles for zoomed region. **(e)** Zoom into marked area of (b).

introduce so-called quadrupoles. A quadrupole consists of a 2×2 block of pixels. The line which should be represented by it is assumed to pass through its centre. Two of the pixels, namely the ones lying completely to the left and the right side of the line, take the minimal and maximal grey value, i.e. black and white. The other two pixels, which are split by the line, are shaded in grey corresponding to the fractions of black and white within those pixels. Examples illustrating such quadrupoles are shown in Fig. 11(d).
A shape as depicted in Fig. 11(a) can now be encoded by placing quadrupoles along its contour. Since quadrupoles on the one hand encode lines, EED on the other hand has the tendency to reduce the curvature, we should set more quadrupoles at contour regions with higher curvature. Therefore, we suggest the following algorithm to distribute the quadrupoles along the contour: We first compute the magnitude of the curvature,

$$|\kappa(u)| = \left| \frac{u_x^2 u_{yy} - 2u_x u_y u_{xy} + u_y^2 u_{xx}}{|\boldsymbol{\nabla} u|^3} \right| ,$$

in every point along the smoothed shape contour. We rescale the result such that the average curvature value of the contour pixels corresponds to the fraction of pixels which should be kept as quadrupole positions. Then we use one-dimensional error diffusion to binarise the curvature signal along the contour and obtain the positions for our quadrupoles. In this way, we set the number of quadrupoles along the contour in proportion to its curvature. This approach also resembles the method presented in [9], in which the density of the interpolation data is chosen proportionally to the Laplacian magnitude. Note that the density of quadrupoles at corners can be tuned by replacing $|\kappa|$ by $|\kappa|^p$, with $p \in \mathbb{R}^+$. In our example we

use $p = 0.5$.
For each of the selected quadrupoles, we finally need to determine the direction which it should encode. This is done by exploiting the dominant eigenvector of the diffusion tensor (see equation (12)) at the centre of the quadrupole. It is pointing in the direction of highest contrast within some neighbourhood, i.e. across the contour. Therefore, by choosing the orthogonal direction, it allows a robust estimation of the tangential direction of the contour at the corresponding location.
Figure 11(d) shows a zoom into the sparse version of the original shape (see Fig. 11(a)) when applying the afore mentioned method. The reconstruction using EED interpolation is depicted in Fig. 11(b).
Obviously, EED is able to encode arbitrary shapes by exploiting simple concepts as provided by quadrupoles. This emphasises once more the excellent interpolation properties of EED. Exploring the shape coding properties of EED in more detail is part of our ongoing research.

6.2 3-D Data

A simple method often used to compress 3-D images is to treat them as a sequence of 2-D image slices. Each slice is then compressed using an established 2-D compression algorithm. However, this approach results in unnecessary overhead and ignores significant potentials for improvement by exploiting redundancies that arise due to the additional dimension. Our 3-D codec C-EED differs from this naive approach in four key aspects:
First of all, file headers are largely redundant for most of the slices. For example, the image dimension is always the same. This overhead can be easily avoided by defining a novel 3-D file format that eliminates header redundancy. Thus, our header has the same structure as in the 2-D case (see Section 4.9), except for the additional image size value which has to be stored for the third dimension.
Secondly, entropy coders typically handle large, cohesive data blocks more efficiently than individually coded segments. Therefore, we use global entropy coding of the whole image data instead of treating slices separately.
Thirdly, we use a 3-D version of EED instead of performing 2-D diffusion in each slice to improve the interpolation quality. Even though the third dimension of the original image may have been sampled coarser (e.g. for medical image data), it still offers valuable information that can be exploited for more efficient inpainting. The mathematical theory behind EED in 3-D is very similar to the 2-D case: The only difference is that two eigenvalues of the smoothed diffusion tensor are set to 1 instead of only one [55].
Finally, the rectangular subdivision scheme used in R-EED is naturally extended to 3-D by replacing rectangles by their 3-D counterpart, i.e. cuboids. Analo-

gously to the 2-D case, the image is split in the middle of its largest dimension, which yields two cuboidal subimages. Thus, the binary tree structure from the two-dimensional case is preserved. Again, many different point patterns can be defined. According to our experiments, using the centre and corners of each 3-D subimage yields very good results. This choice of points can be regarded as the natural 3-D extension of the 2-D point pattern F (see Fig. 6).
In the 3-D setting, we use real world medical data acquired by computerised tomography (CT) to evaluate the performance of our 3-D codec C-EED. We first compare C-EED to a modified version of R-EED to assess the influence of the 3-D diffusion on compression quality. The only difference between R-EED and C-EED is the application of 2-D EED to image slices (R-EED) instead of 3-D diffusion on the whole image (C-EED). Other influences such as the effects of global entropy coding and header redundancies are avoided. As expected, the 3-D diffusion offers a significant advantage over its 2-D counterpart (see Fig. 12).
Furthermore, we compare our results to a widely-used transformation-based image coding standard for medical images, the DICOM standard [52]. In particular, we apply the most efficient compression standard allowed by DICOM, namely JPEG 2000. Just as for R-EED, we eliminate header redundancies and disadvantages due to slice-wise entropy coding by applying JPEG 2000 to a single 2-D image that contains all slices. As the 2-D experiments with R-EED already suggest, C-EED performs significantly better than DICOM.

7 Conclusion

In this article, we have contributed to the advancement of PDE-based image compression in three ways:
First of all, we have gained a substantially deeper understanding why the edge-enhancing anisotropic diffusion (EED) operator is ideally suited for interpolating missing data in compression applications: Its anisotropy in conjunction with its semilocal behaviour offers specific advantages for this task. It can create edges that are smooth along the edge, but permits contrast jumps across the edge. This allows to model realistic edge contours by specifying only a very small number of pixels. By avoiding singularities at interpolation points as well as over- and undershoots near edges, it combines distinctive advantages of second and higher order interpolation operators.
Our second contribution consists of an improved EED-based codec that outperforms not only the recent EED-based codec from Galić et al. [33], but also sophisticated compression standards such as JPEG 2000. Its high quality is caused by a careful optimisation of a number of intermediate steps, such as rectangular subdivision, selection of diffusion parameters, adaptation of the brightness values,

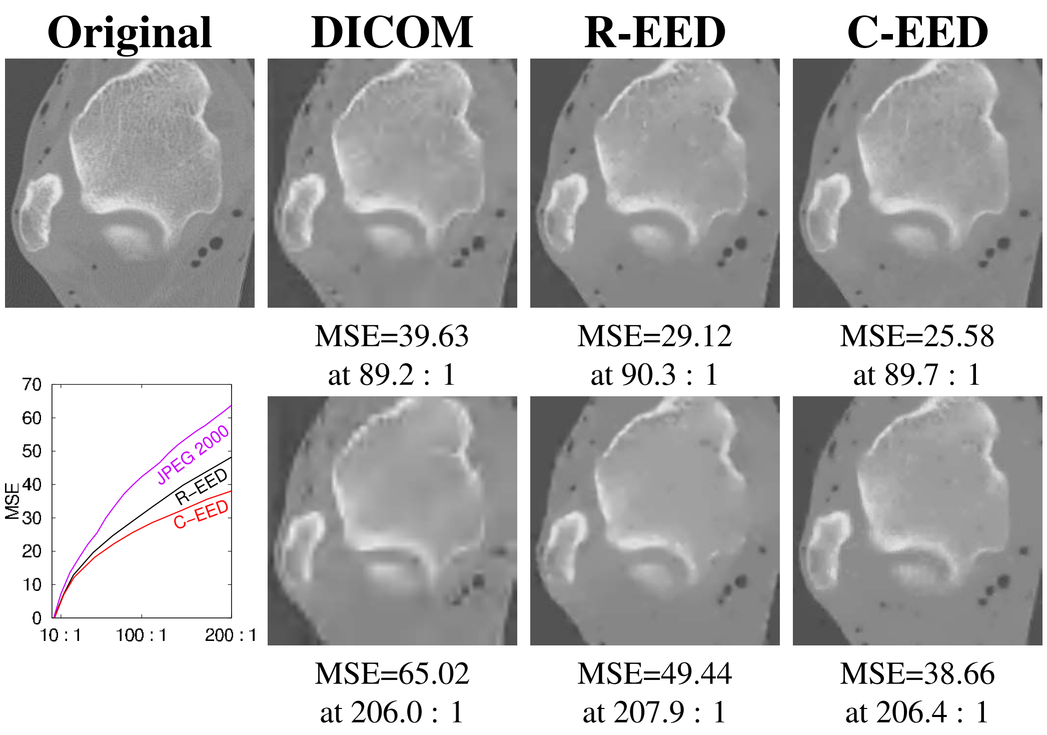


Figure 12: Images obtained with DICOM (using JPEG 2000), R-EED, and C-EED with compression rates close to 90:1 (top row) and 207:1 (bottom row). The test data set “trab64” (size $256 \times 256 \times 64$) consists of the first 64 slices of a femured bone CT. The images above depict the last 2-D slice of “trab64”.

entropy encoding, and interpolation swapping.
Thirdly, we have demonstrated that EED-based compression is also applicable for shape coding, and that the concepts of our EED-based codec translate well to the 3-D setting. Especially, we showed that the use of 3-D EED offers significant advantages over slice-wise 2-D diffusion, and that our 3-D codec C-EED outperforms R-EED as well as the DICOM/JPEG 2000 standard on medical data.
It is evident that the basic ideas behind EED-based image compression generalise in a straightforward way to tensor data sets, if one uses the corresponding EED operators from [70]. It is more challenging to adapt our methods to the efficient compression of image sequences and surface data, see e.g. [6]. A more advanced handling of colour images and shapes is also on our agenda. Moreover, we are going to focus on approaches that pay specific attention to highly textured images. On an algorithmic side, we are currently evaluating a number of efficient numerical methods, and we are also studying parallel implementations on multi-core architectures such as GPUs.

Acknowledgements

We thank Irena Galić (University of Osijek, Croatia) for fruitful discussions and for providing two images and code of her compression algorithm, David Tschumperlé (French National Center for Scientific Research, France) for providing the image in Fig. 1, and Wiro Niessen (University Medical Center Rotterdam, The Netherlands) for providing the test image in Fig. 12.

References

[1] T. Acar and M. Gökmen. Image coding using weak membrane model of images. In A. K. Katsaggelos, editor, *Visual Communications and Image Processing '94*, volume 2308 of *Proceedings of SPIE*, pages 1221–1230. SPIE Press, Bellingham, 1994.

[2] F. Alter, S. Durand, and J. Froment. Adapted total variation for artifact free decompression of JPEG images. *Journal of Mathematical Imaging and Vision*, 23(2):199–211, Sept. 2005.

[3] H. A. Aly and E. Dubois. Image up-sampling using total-variation regularization with a new observation model. *IEEE Transactions on Image Processing*, 14(10):1647–1659, Oct. 2005.

[4] G. Aronsson. Extension of functions satisfying Lipschitz conditions. *Arkiv för Matematik*, 6(6):551–561, June 1967.

[5] V. Aurich and U. Daub. Bilddatenkompression mit geplanten Verlusten und hoher Rate. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Mustererkennung 1996*, pages 138–146. Springer, Berlin, 1996.

[6] E. Bae and J. Weickert. Partial differential equations for interpolation and compression of surfaces. In M. Daehlen, M. Floater, T. Lyche, J.-L. Merrien, K. Mørken, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 1–14. Springer, Berlin, 2010.

[7] S. Battiato, G. Gallo, and F. Stanco. Smart interpolation by anisotropic diffusion. In *Proc. Twelvth International Conference on Image Analysis and Processing*, pages 572–577. IEEE Computer Society Press, Montova, Italy, Sept. 2003.

[8] A. Belahmidi and F. Guichard. A partial differential equation approach to image zoom. In *Proc. 2004 IEEE International Conference on Image Processing*, volume 1, pages 649–652. Singapore, Oct. 2004.

[9] Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert. How to choose interpolation data in images. *SIAM Journal on Applied Mathematics*, 70(1): 333–352, 2009.

[10] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. SIGGRAPH 2000*, pages 417–424. New Orleans, LI, July 2000.

[11] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, July 2007.

[12] A. Borzì, H. Grossauer, and O. Scherzer. Analysis of iterative methods for solving a Ginzburg-Landau equation. *International Journal of Computer Vision*, 64(2–3):203–219, 2005.

[13] S. Bougleux, G. Peyré, and L. Cohen. Image compression with anisotropic triangulations. In *Proc. Tenth International Conference on Computer Vision*. Kyoto, Japan, Oct. 2009.

[14] P. Bourdon, B. Augereau, C. Chatellier, and C. Olivier. MPEG-4 compression artifacts removal on color video sequences using 3D nonlinear diffusion. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 729–732. IEEE Computer Society Press, Montreal, Canada, Mai 2004.

[15] R. Bourne. *Fundamentals of Digital Imaging in Medicine*. Springer, London, 2010.

[16] A. M. Bruckstein. On image extrapolation. Technical Report CIS9316, Computer Science Department, Technion, Haifa, Israel, Apr. 1993.

[17] A. Bruhn and J. Weickert. A confidence measure for variational optic flow methods. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Geometric Properties from Incomplete Data*, volume 31 of *Computational Imaging and Vision*, pages 283–297. Springer, Dordrecht, 2006.

[18] E. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, Feb. 2006.

[19] S. Carlsson. Sketch based coding of grey level images. *Signal Processing*, 15:57–83, 1988.

[20] V. Caselles, J.-M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3):376–386, Mar. 1998.

[21] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 32:1895–1909, 1992.

[22] T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (CDD). *Journal of Visual Communication and Image Representation*, 12(4): 436–449, 2001.

[23] T. F. Chan and H. M. Zhou. Total variation improved wavelet thresholding in image compression. In *Proc. Seventh International Conference on Image Processing*, volume II, pages 391–394. Vancouver, Canada, Sept. 2000.

[24] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311, 1997.

[25] L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[26] U. Y. Desai, M. M. Mizuki, I. Masaki, and B. K. P. Horn. Edge and mean based image compression. Technical Report 1584 (A.I. Memo), Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, U.S.A., Nov. 1996.

[27] M. Dipperstein. Michael Dipperstein's page o'stuff, January 2009. URL `http://michael.dipperstein.com/index.html`.

[28] R. Distasi, M. Nappi, and S. Vitulano. Image compression by B-tree triangular coding. *IEEE Transactions on Communications*, 45(9):1095–1100, Sept. 1997.

[29] J. H. Elder. Are edges incomplete? *International Journal of Computer Vision*, 34(2/3):97–122, 1999.

[30] G. Facciolo, F. Lecumberry, A. Almansa, A. Pardo, V. Caselles, and B. Rougé. Constrained anisotropic diffusion and some applications. In *Proc. 2006 British Machine Vision Conference*, volume 3, pages 1049–1058. Edinburgh, Scotland, Sept. 2006.

[31] G. E. Ford. Application of inhomogeneous diffusion to image and video coding. In *Proc. 13th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 926–930. Asilomar, CA, Nov. 1996.

[32] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Towards PDE-based image compression. In N. Paragios, O. Faugeras, T. Chan, and C. Schnörr, editors, *Variational, Geometric and Level-Set Methods in Computer Vision*, volume 3752 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Berlin, 2005.

[33] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2–3):255–269, July 2008.

[34] A. R. Gourlay. Hopscotch: a fast second-order partial differential equation solver. *IMA Journal of Applied Mathematics*, 6(4):375–390, 1970.

[35] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.

[36] R. Hummel and R. Moniot. Reconstructions from zero-crossings in scale space. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37: 2111–2130, 1989.

[37] T. Iijima. Basic theory of pattern observation. In *Papers of Technical Group on Automata and Automatic Control*. IECE, Japan, Dec. 1959. In Japanese.

[38] P. Johansen, S. Skelboe, K. Grue, and J. D. Andersen. Representing signals by their toppoints in scale space. In *Proc. Eighth International Conference on Pattern Recognition*, pages 215–217. Paris, France, Oct. 1986.

[39] F. M. W. Kanters, M. Lillholm, R. Duits, B. J. P. Jansen, B. Platel, L. Florack, and B. M. ter Haar Romeny. On image reconstruction from multiscale top points. In R. Kimmel, N. Sochen, and J. Weickert, editors, *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 431–439. Springer, Berlin, 2005.

[40] I. Kopilovic and T. Szirányi. Artifact reduction with diffusion preprocessing for image compression. *Optical Engineering*, 44(2):1–14, Feb. 2005.

[41] H. Köstler, M. Stürmer, C. Freundl, and U. Rüde. PDE based video compression in real time. Technical Report 07-11, Lehrstuhl für Informatik 10, Univ. Erlangen–Nürnberg, Germany, 2007.

[42] M. Kunt, A. Ikonomopoulos, and M. Kocher. Second-generation image-coding techniques. *Proceedings of the IEEE*, 73(4):549–574, Apr. 1985.

[43] M. Lillholm, M. Nielsen, and L. D. Griffin. Feature-based image analysis. *International Journal of Computer Vision*, 52(2/3):73–95, 2003.

[44] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. Image compression with edge-based inpainting. *IEEE Transactions on Circuits, Systems and Video Technology*, 17(10):1273–1286, Oct. 2007.

[45] M. Mahoney. Adaptive weighing of context models for lossless data compression. Technical Report CS-2005-16, Florida Institute of Technology, Melbourne, Florida, Dec. 2005.

[46] M. Mahoney. Data compression programs, November 2009. URL `http://mattmahoney.net/dc/`.

[47] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann, and B. Doerr. Optimising spatial and tonal data for homogeneous diffusion inpainting. In A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 26–37. Springer, Berlin, June 2012.

[48] M. Mainberger and J. Weickert. Edge-based image compression with homogeneous diffusion. In X. Jiang and N. Petkov, editors, *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 476–483. Springer, Berlin, 2009.

[49] F. Malgouyres and F. Guichard. Edge direction preserving image zooming: A mathematical and numerical analysis. *SIAM Journal on Numerical Analysis*, 39(1):1–37, 2001.

[50] S. Mallat and S. Zhong. Characterisation of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:720–732, 1992.

[51] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 259–263. Chicago, IL, Oct. 1998.

[52] National Electrical Manufacturers Association. Digital Imaging and Communications in Medicine (DICOM) – Part 5 Data Structures and Encoding. PS 3.5-2004, 2004.

[53] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Springer, New York, 1992.

[54] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[55] P. Peter. Three-dimensional data compression with anisotropic diffusion. In *Proc. DAGM-OAGM 2012 Symposium for Pattern Recognition, Young Researchers Forum*. Springer, Berlin, 2012.

[56] S. D. Rane, G. Sapiro, and M. Bertalmio. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. *IEEE Transactions on Image Processing*, 12(3):296–302, Mar. 2003.

[57] J. J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

[58] A. Roussos and P. Maragos. Vector-valued image interpolation by an anisotropic diffusion-projection PDE. In F. Sgallari, F. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 104–115. Springer, Berlin, 2007.

[59] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 452–461. Springer, Berlin, 2009.

[60] A. Solé, V. Caselles, G. Sapiro, and F. Arandiga. Morse description and geometric encoding of digital elevation maps. *IEEE Transactions on Image Processing*, 13(9):1245–1262, Sept. 2004.

[61] P. Strobach. Quadtree-structured recursive plane decomposition coding of images. *IEEE Transactions on Signal Processing*, 39(6):1380–1397, June 1991.

[62] G. J. Sullivan and R. J. Baker. Efficient quadtree coding of images and video. *IEEE Transactions on Image Processing*, 3(3):327–331, May 1994.

[63] D. S. Taubman and M. W. Marcellin, editors. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer, Boston, 2002.

[64] A. Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004.

[65] D. Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's. *International Journal of Computer Vision*, 68 (1):65–82, June 2006.

[66] D. Tschumperlé and R. Deriche. Vector-valued image regularization with PDEs: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):506–516, Apr. 2005.

[67] H. Tsuji, T. Sakatani, Y. Yashima, and N. Kobayashi. A nonlinear spatio-temporal diffusion and its application to prefiltering in MPEG-4 video coding. In *Proc. 2002 IEEE International Conference on Image Processing*, volume 1, pages 85–88. Rochester, NY, Sept. 2002.

[68] H. Tsuji, S. Tokumasu, H. Takahashi, and M. Nakajima. Spatial prefiltering scheme based on anisotropic diffusion in low-bitrate video coding. *Systems and Computers in Japan*, 38(10):34–45, 2007.

[69] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing Supplement*, 11:221–236, 1996.

[70] J. Weickert. Nonlinear diffusion filtering. In B. Jähne, H. Haußecker, and P. Geißler, editors, *Handbook on Computer Vision and Applications, Vol. 2: Signal Processing and Pattern Recognition*, pages 423–450. Academic Press, San Diego, 1999.

[71] J. Weickert and T. Brox. Diffusion and regularization of vector- and matrix-valued images. In M. Z. Nashed and O. Scherzer, editors, *Inverse Problems, Image Analysis, and Medical Imaging*, volume 313 of *Contemporary Mathematics*, pages 251–268. AMS, Providence, 2002.

[72] J. Weickert, S. Ishikawa, and A. Imiya. Linear scale-space has first been proposed in Japan. *Journal of Mathematical Imaging and Vision*, 10(3): 237–252, May 1999.

[73] J. Weickert and M. Welk. Tensor field interpolation with PDEs. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields*, pages 315–325. Springer, Berlin, 2006.

[74] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[75] Y. Wu, H. Zhang, Y. Sun, and H. Guo. Two image compression schemes based on image inpainting. In *Proc. 2009 International Joint Conference on Computational Sciences and Optimization*, pages 816–820. IEEE Computer Society Press, Apr. 2009.

[76] Z. Xie, W. R. Franklin, B. Cutler, M. A. Andrade, M. Inanc, and D. M. Tracy. Surface compression using over-determined Laplacian approximation. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations XVII*, volume 6697 of *Proceedings of SPIE*. SPIE Press, Bellingham, 2007.

[77] Z. W. Xiong, X. Y. Sun, F. Wu, and S. P. Li. Image coding with parameter-assistant inpainting. In *Proc. 2007 IEEE International Conference on Image Processing*, volume 2, pages 369–372. San Antonio, TX, Sept. 2007.

[78] Y. Zeevi and D. Rotem. Image reconstruction from zero-crossings. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34:1269–1277, 1986.