# Universität des Saarlandes

# Fachrichtung 6.1 – Mathematik

Preprint Nr. 203

**Image Compression with Anisotropic Diffusion**

Irena Galić, Joachim Weickert, Martin Welk,
Andrés Bruhn, Alexander Belyaev and
Hans-Peter Seidel

Saarbrücken 2008

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 203
submitted: January 7, 2008

# Image Compression with Anisotropic Diffusion

**Irena Galić**
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Saarland University
Campus E1.1, 66041 Saarbrücken, Germany
galic@mia.uni-saarland.de

**Joachim Weickert**
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Saarland University
Campus E1.1, 66041 Saarbrücken, Germany
weickert@mia.uni-saarland.de

**Martin Welk**
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Saarland University
Campus E1.1, 66041 Saarbrücken, Germany
welk@mia.uni-saarland.de

**Andrés Bruhn**
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Saarland University
Campus E1.1, 66041 Saarbrücken, Germany
bruhn@mia.uni-saarland.de

**Alexander Belyaev**
EECE, School of Engineering and Physical Sciences
Earl Mountbatten Building, Heriot-Watt University
Riccarton, Edinburgh EH14 4AS, UK
a.belyaev@hw.ac.uk

**Hans-Peter Seidel**
Max-Planck Institute for Informatics
Campus E1.4, 66123 Saarbrücken, Germany
hpseidel@mpi-sb.mpg.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/ |

**Abstract**

Compression is an important field of digital image processing where well-engineered methods with high performance exist. Partial differential equations (PDEs), however, have not much been explored in this context so far. In our paper we introduce a novel framework for image compression that makes use of the interpolation qualities of edge-enhancing diffusion. Although this anisotropic diffusion equation with a diffusion tensor was originally proposed for image denoising, we show that it outperforms many other PDEs when sparse scattered data must be interpolated. To exploit this property for image compression, we consider an adaptive triangulation method for removing less significant pixels from the image. The remaining points serve as scattered interpolation data for the diffusion process. They can be coded in a compact way that reflects the B-tree structure of the triangulation. We supplement the coding step with a number of amendments such as error threshold adaptation, diffusion-based point selection, and specific quantisation strategies. Our experiments illustrate the usefulness of each of these modifications. They demonstrate that for high compression rates, our PDE-based approach does not only give far better results than the widely-used JPEG standard, but can even come close to the quality of the highly optimised JPEG2000 codec.

# 1 Introduction

While applications of partial differential equations (PDEs) and corresponding variational techniques in image processing and computer vision are often associated with denoising problems (see e.g. [3, 17, 68]), there is an increasing number of publications where their potential as interpolation methods is explored.

Early examples in that direction include variational optic flow methods [35, 54] where one is interested in estimating the apparent motion field in an image sequence as a minimiser of an energy functional. In flat regions where the local greyscale data do not allow to compute reliable motion fields, information from surrounding regions is propagated by the so-called *filling-in effect* of the smoothness term. Similar variational models are also used for related correspondence problems such as stereo reconstruction [49, 74] or image registration [6, 52], whenever dense displacement fields are required.

More recently this filling-in effect has also become the main feature of PDE-based inpainting methods such as [10, 11, 16, 34, 50, 65]. Here one aims at restoring missing informations in certain corrupted image areas by means of second or higher-order PDEs. The basic idea is to regard the given image data

as Dirichlet boundary conditions, and interpolate the data in the inpainting regions by solving appropriate boundary value problems.
Variational and PDE methods that have been proposed for inpainting have also been investigated for more classical interpolation problems such as zooming into an image by increasing its resolution [2, 7, 8, 15, 47, 60, 69]. For such interpolation problems with data given on a regular grid, these techniques are in competition with cubic or quintic splines, radial basis functions and sinc-based interpolation techniques; see e.g. [44, 51]. If the data are not available on a regular grid, scattered data interpolation techniques have been proposed [31, 55], among which radial basis functions such as thin plate splines [25] are popular and well-performing.
Interestingly, not many of the variational and PDE-based interpolation and inpainting techniques have been used for scattered data interpolation. It seems that the sparsity of the scattered data constitutes a real challenge for these techniques: While second-order PDEs may satisfy a maximum–minimum principle, they often create singularities at isolated interpolation points in 2-D. Higher-order PDEs, on the other hand, may give smoother solutions, but the violation of an extremum principle can lead to undesirable over- and undershoots; see e.g [15].

The goal of the present paper is to explore the potential of PDEs for a difficult scattered data interpolation problem, namely lossy image compression. While contemporary image compression methods are dominated by concepts that involve the discrete cosine transform (such as the widely-used JPEG standard [56]) or the discrete wavelet transform (in JPEG2000 [64]), we shall give a proof-of-concept that there are alternatives where PDEs may be beneficial. The basic idea is to reduce the image data to a well-adapted set of significant sparse points that can be coded in an efficient way. Decoding is accomplished by using these scattered data and interpolating them by means of a suitable PDE. Our PDE of choice is *edge-enhancing anisotropic diffusion (EED)*. It uses a diffusion tensor that allows smoothing along discontinuities while inhibiting smoothing across them. Although EED has been introduced originally as a denoising technique [67], we will see that it is particularly useful for scattered data interpolation. As a tool for creating a useful sparse point representation, we consider the *B-tree triangular coding (BTTC)* by Distasi et al. [24], since it is relatively simple and allows an efficient coding of the sparsified image data. In order to end up with a PDE-based compression framework with optimal quality, however, it is not sufficient to apply BTTC in its original version. It has to be supplemented with a number of amendments such as error threshold adaptation, diffusion-based point selection and specific quantisation strategies.

**Paper Structure.** Our paper is organised as follows. In Section 2 we describe PDE-based interpolation techniques and show that scattered data interpolation with EED performs particularly well. In Section 3 we review the BTTC model of Distasi et al. [24] for image coding and decoding. Section 4 describe how it can be coupled with EED-based interpolation, and presents a number of amendments for the coding step. Experiments on EED-based image coding are presented in Section 5, where each of these amendments is demonstrated to lead to improved compression quality. Our paper is concluded with a summary in Section 6.

**Related Work.** In the context of image compression, PDEs and related variational techniques have mainly been used as a preprocessing step before coding images or videos [18, 30, 29, 41, 66] or as a postprocessing tool for removing coding artifacts [1, 26, 29, 33, 53, 72, 73]. Our work differs from these papers by the fact that we use a PDE *within* the encoding and decoding step rather than applying it before encoding or after decoding. Chan and Zhou [19] proposed total variation regularisation in order to modify the coefficients in a wavelet decomposition such that oscillatory edge artifacts are reduced. Sometimes PDE-based interpolation strategies have been tailored to specific data sets such as surface data in digital elevation maps [28, 61, 70]. Moreover, some variational $L^1$ minimisation ideas play an important role in recent compressed sensing concepts [13].

The usefulness of inpainting concepts for image compression is confirmed in several papers, where structure and texture inpainting ideas have been integrated into standard codecs (i.e. compression and decompression methods) such as JPEG [46, 58, 71].

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 different formulations by Zeevi and Rotem [75], Carlsson [14], Hummel and Moniot [37], Mallat and Zhong [48], Aurich and Daub [4], Desai et al. [23], and Elder [27]. Methods of this type can be seen as representatives of second-generation coding approaches that exploit perceptually relevant features such as contours [43].

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 Johansen et al. [39] and Kanters et al. [40]. 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 Lillholm et al. [45]. Impressive global reconstructions of natural images by means of the local jet structure are reported by Bruckstein [12], who suggested to incorporate Tikhonov regularisation and directional filtering.

With respect to the triangular B-tree coding of Distasi et al. [24] that is used in our paper, a number of related variable block size image coding algorithms exist, in particular also methods based on quadtree decomposions; see e.g. [62, 63]. An interesting coding scheme that exploits scattered data interpolation has been proposed by Demaret et al. [22]. They construct an adaptive Delaunay triangulation that is used for decoding the image by linear interpolation. Their experiments show that it can be an alternative to JPEG 2000 coding for texture-free images.

In 2005 a preliminary version of our paper has been presented at a workshop [32]. Compared to this earlier model, the present paper introduces numerous improvements in Section 4, it presents a systematic evaluation of all these modifications, and it compares the results with JPEG2000 for a set of six standard test imags. Recently Köstler et al. [42] have developped multigrid algorithms for our method from [32] and demonstrated that they can be used for real-time video compression on a *Playstation 3*.

# 2 PDE-Based Interpolation

We start by considering a PDE approach to image interpolation. First we discuss a general model, then we investigate a number of options for smoothing operators, and finally we present an experiment that illustrates their performance.

## 2.1 General Model

Let $\Omega \subset \mathbb{R}^n$ denote an $n$-dimensional image domain. We want to recover some unknown scalar-valued function $v : \Omega \to \mathbb{R}$, from which we know its values on some subset $\Omega_1 \subset \Omega$. Our goal is to find an interpolating function $u : \Omega \to \mathbb{R}$ that is smooth and close to $v$ in $\Omega \setminus \Omega_1$ and identical to $v$ in $\Omega_1$. We may embed this problem in an evolution setting with some evolution parameter (the "time") $t \geq 0$. It solution $u(x,t)$ gives the desired interpolating function as its steady state ($t \to \infty$). We initialise the evolution with some function $f : \Omega \to \mathbb{R}$ that is identical to $v$ on $\Omega_1$ and that is set to some arbitrary value (e.g. to 0) on $\Omega \setminus \Omega_1$:

$$f(x) := \begin{cases} v(x) & \text{if } x \in \Omega_1 \\ 0 & \text{else.} \end{cases} \tag{1}$$

We consider the evolution

$$\partial_t u \;=\; (1-c(x))\, Lu \;-\; c(x)\,(u-f) \tag{2}$$

with $f$ as initial value,

$$u(x,0) \;=\; f(x), \tag{3}$$

and reflecting (homogeneous Neumann) boundary conditions on the image boundary $\partial\Omega$. The function $c : \Omega \to \mathbb{R}$ is the characteristic function on $\Omega_1$, i.e.

$$c(x) := \begin{cases} 1 & \text{if } x \in \Omega_1 \\ 0 & \text{else,} \end{cases} \tag{4}$$

and $L$ is some elliptic differential operator. The idea is to solve the steady state equation

$$(1-c(x))\, Lu \;-\; c(x)\,(u-f) = 0 \tag{5}$$

with reflecting boundary conditions. In $\Omega_1$ we have $c(x) = 1$ such that the interpolation condition $u(x) = f(x) = v(x)$ is fulfilled. In $\Omega \setminus \Omega_1$ it follows from $c(x) = 0$ that the solution has to satisfy $Lu = 0$. This elliptic PDE can be regarded as the steady state of the evolution equation

$$\partial_t u = Lu \tag{6}$$

with Dirichlet boundary conditions given by the interpolation data on $\Omega_1$.

## 2.2 Specific Smoothing Operators

Regarding the elliptic differential operator $L$, many possibilities exist. The simplest one uses the Laplacian $Lu := \Delta u$ leading to *homogeneous diffusion* [38]:

$$\partial_t u = \Delta u. \tag{7}$$

A prototype for a higher order differential operator is the biharmonic operator $Lu := -\Delta^2 u$ giving the *biharmonic smoothing* evolution

$$\partial_t u = -\Delta^2 u. \tag{8}$$

Using it for interpolation comes down to thin plate spline interpolation [25], a rotationally invariant multidimensional generalisations of cubic spline interpolation.

The multidimensional generalisation of quintic spline interpolation leads to *triharmonic smoothing* based on $Lu := \Delta^3 u$:

$$\partial_t u = \Delta^3 u. \tag{9}$$

Note that only the second-order differential operators allow a maximum–minimum principle, where the values of $u$ stay within the range of the values of $f$ in $\Omega_1$.

A second order PDE that has been advocated for interpolation purposes [15] is given by the *absolute monotone Lipschitz extension (AMLE)* model. It uses the second order directional derivative $Lu := \partial_{\eta\eta} u$ in gradient direction $\eta := \nabla u / |\nabla u|$:

$$\partial_t u = \partial_{\eta\eta} u. \tag{10}$$

*Nonlinear isotropic diffusion* processes are governed by the equation $Lu := \operatorname{div}\left(g(|\nabla u|^2)\, \nabla u\right)$. This gives [57]

$$\partial_t u = \operatorname{div}\left(g(|\nabla u|^2)\, \nabla u\right) \tag{11}$$

where the diffusivity function $g$ is decreasing in its argument, since the goal is to reduce smoothing at edges where $|\nabla u|$ is large. One may e.g. choose the *Charbonnier diffusivity* [20]

$$g(s^2) = \frac{1}{\sqrt{1 + s^2/\lambda^2}} \tag{12}$$

with some contrast parameter $\lambda > 0$. Since (11) uses a scalar-valued diffusivity we name this process *isotropic* (in contrast to the nomenclature in [57]).

Real anisotropic behaviour is possible when a diffusion tensor is used. As a prototype for nonlinear anisotropic diffusion filtering we consider *edge-enhancing diffusion (EED)* [67]. The idea is to reduce smoothing across edges while still permitting diffusion along them. The EED diffusion tensor has one eigenvector parallel to $\nabla u_\sigma$, where $u_\sigma$ is obtained from convolving $u$ with a Gaussian with standard deviation $\sigma$. The associated eigenvalue is given by $g(|\nabla u_\sigma|^2)$ with a diffusivity function such as (12). The other eigenvectors are orthogonal to $\nabla u_\sigma$ and have corresponding eigenvalues 1. If we use the convention to extend a scalar-valued function $g(x)$ to a matrix-valued function $g(A)$ by applying $g$ to the eigenvalues on $A$ and leaving the eigenvectors unchanged, then EED can be formally linked to $Lu := \operatorname{div}\left(g(\nabla u_\sigma \nabla u_\sigma^\top)\, \nabla u\right)$. Hence, its evolution is governed by

$$\partial_t u = \operatorname{div}\left(g(\nabla u_\sigma \nabla u_\sigma^\top)\, \nabla u\right). \tag{13}$$

## 2.3 Experiments on Interpolation

In order to evaluate the potential of the preceding PDEs for scattered data interpolation, we have discretised them with central finite differences in space. For the diffusion equations, we performed a semi-implicit time discretisation with SOR as solver for the linear systems of equations, while for AMLE, biharmonic and triharmonic smoothing an explicit scheme was used. Runtimes for a non-optimised C implementation on a 1.5 GHz Centrino laptop range between a few seconds and several minutes for a $256 \times 256$ image. If necessary, these operations can be speeded up to real-time performance using multigrid algorithms [42].

In Figure 1 we present an experiment that illustrates the use of the different smoothing operators for scattered data interpolation. It depicts a zoom into the famous *lena* image, where 2 percent of all pixels have been chosen randomly as scattered interpolation points. We observe that homogeneous diffusion is not very suitable for scattered data interpolation, since it creates singularities at the interpolation points. This can be avoided with interpolation using biharmonic smoothing. It gives fairly good results, but suffers from over- and undershoots near edges due to the violation of an extremum principle (see e.g. the shoulder). These limitations become even more obvious for triharmonic smoothing. In spite of a number of favourable theoretical properties [15], AMLE does not live up to its expectations. Also going from homogeneous diffusion to nonlinear isotropic diffusion does not give an improvement: While nonlinear isotropic diffusion may allow discontinuities, its interpolant is too flat and tends to keep many interpolation points as isolated singularities. The fact that EED, on the other hand, gives the best results shows the importance of the anisotropic behaviour. Its ability to smooth along edges seems to be very beneficial for avoiding singularities at interpolation points. Moreover, this second-order PDE respects a maximum–minimum principle, such that the solution is within the greyscale bounds of the interpolation points.

A quantitative error analysis is presented in Table 1, where we have computed the *average absolute error (AAE)* and the *mean squared error (MSE)* between the interpolated image $(u_{i,j})$ and the original image $(v_{i,j})$:

$$\mathrm{AAE}\,(u,v) := \frac{1}{N}\sum_{i,j} |u_{i,j} - v_{i,j}|, \tag{14}$$

$$\mathrm{MSE}\,(u,v) := \frac{1}{N}\sum_{i,j} |u_{i,j} - v_{i,j}|^2, \tag{15}$$

Figure 1: **(a) Top left:** Zoom into the test image *lena*, $256 \times 256$ pixels. **(b) Top middle:** Grey values of the scattered interpolation points (2 percent of all pixels, chosen randomly). **(c) Middle left:** Interpolation by linear diffusion. **(d) Centre:** Biharmonic smoothing. **(e) Middle right:** Triharmonic smoothing. **(f) Bottom left:** AMLE. **(g) Bottom middle:** Nonlinear isotropic diffusion ($\lambda = 0.1$). **(h) Bottom right:** EED ($\lambda = 0.1$, $\sigma = 1$).

Table 1: Average absolute errors (AAE) and mean squared error (MSE) for the PDEs used for scattered data interpolation in Figure 1.

| PDE method | AAE | MSE |
|---|---|---|
| homogeneous diffusion (7) | 16.98 | 611.5 |
| biharmonic smoothing (8) | 15.79 | 615.5 |
| triharmonic smoothing (9) | 18.69 | 807.9 |
| AMLE (10) | 17.33 | 631.7 |
| Charbonnier diffusion (11) | 21.80 | 987.0 |
| edge-enhancing diffusion (13) | 14.58 | 591.7 |

where $N$ denotes the number of pixels. As already mentioned, biharmonic and triharmonic interpolation may create over- and undershoots. At locations where they lead to grey values outside the interval $[0, 255]$, they have been truncated.

In our experiments the average absolute error gives a ranking that corresponds well with our visual impression, while the mean squared error fails to discriminate between the perceptually strongly differing qualities of linear diffusion, biharmonic smoothing and AMLE. It seems that the MSE puts more weight on deviations of individual pixels than our human perception does. Thus, it may be less helpful for quantifying the perceived visual quality. Interestingly, in both error measurements as well as in our visual analysis, EED gives the best scattered data interpolation. These findings are also in accordance with results from [69] where EED proved to be the PDE of choice for interpolating tensor data. Therefore, we will focus on EED from now on, and all optimisations of the approximation quality will be carried out with respect to the average absolute error.

# 3 Image Coding and Decoding by Binary Trees

Now that we have seen that EED is useful for scattered data interpolation, we want to exploit this technique for image compression. To this end we have to combine it with a method that provides a useful sparse image representation with scattered data. Intuitively the density of these data points should be adapted to the underlying image structure in order to allow the best interpolation quality. We have started to explore a number of options, ranging from semantically motivated feature selection [76] over greedy optimisation concepts [21] to shape optimisation strategies [9]. For compression purposes, however, good interpolation quality is not sufficient if the resulting set of image data is too expensive to encode. This requirement has led us to an image compression and decompression scheme that relies on an adaptive sparsification of the image data by means of the triangulation from *B-tree triangular coding (BTTC)*. In this section we review the underlying codec by Distasi et al. [24]. It serves as starting point for inclusion of our EED-based interpolation concept.

## 3.1 Creating Scattered Interpolation Points

In B-tree triangular coding, an image is decomposed into a number of triangular regions such that within each region it can be recovered in sufficient quality by interpolation from the vertices. In our case, all regions are isosceles right triangles. Then the decomposition into triangles is stored in a binary tree structure.

In order to describe the compression procedure, let us assume we have an image $v = (v_{i,j})$ of size $(2^m + 1) \times (2^m + 1)$. Smaller images should be filled up to such a size in a suitable way. Initially, the image is split by one of its diagonals into two triangles. The four image corners $(1, 1)$, $(1, 2^m + 1)$, $(2^m + 1, 1)$ and $(2^m + 1, 2^m + 1)$ are vertices of these two triangles.

To refine this initial configuration, an approximation $(u_{i,j})$ of the image $(v_{i,j})$ is calculated by using only the grey values from the vertices and interpolating the remaining parts of the image.

For the moment, we consider just the simplest interpolation procedure, that is a linear interpolation within each triangle. If the error $e_{i,j} := |u_{i,j} - f_{i,j}|$ satisfies $e_{i,j} \leq \varepsilon$ for all pixels $(i, j)$, with a given tolerance parameter $\varepsilon > 0$, the representation by triangles is considered sufficiently fine. Otherwise, for each pixel $(i, j)$ for which $e_{i,j} > \varepsilon$ holds, the triangle which contains $(i, j)$ is split into two similar triangles by the height on its hypotenuse. The centre of the hypotenuse thereby becomes an additional vertex of the representation. By recalculating approximation errors within the new smaller triangles, it is determined whether to split these again etc. Since the approximation error is zero at vertices, triangles with legs of length 1 are not split further, which guarantees that the recursive splitting terminates. Moreover, vertices throughout the process have integer coordinates. A vertex mask of size equal to the image is generated during the triangulation in order to indicate which pixels are vertices.

One point which needs additional consideration is the treatment of pixels located on the sides of triangles during the splitting process. If the error bound is violated in such a pixel, it is sufficient for our compression and decompression method to split one of the two adjacent triangles. This allows to reduce the number of triangles noticeably since in regions with fine details, a large number of small triangles occur, and many pixel positions then happen to be located on sides.

## 3.2 Coding the Binary Tree

To efficiently store the triangulation, we notice that the hierarchical splitting of triangles gives rise to a binary tree structure. Each triangle occurring during the splitting process is represented by a node while leaves correspond to those triangles which are not divided further. When a triangle is split, its two subtriangles become the children of its representing node. To store the structure of the tree, one traverses the tree and stores one bit per node: a 1 for a node that has children, and a 0 for a leave. Preorder or level-order traversal are equally possible. Note that by the tree structure, the vertex mask is fully determined. Further space in storing the tree is saved by measuring globally the minimal and maximal depth of the tree. Only for nodes at intermediate levels, the corresponding bits are stored.
For coding the grey values in all vertices, we first zig zag traverse the sparse image created with the binary tree structure and store it in a sequence of grey values. Then this sequence is compressed by Huffman coding [36], a lossless variable-length prefix code that assigns shorter codes to more frequent characters. Its code structure is represented by a binary tree, too.
Our entire coded image format then reads as follows:

- image size (4 bytes)
- minimal and maximal depth of the binary tree (together 2 bytes)
- binary string encoding binary tree structure (1 bit for each node between minimal and maximal depth, filled up with zeros to the next byte boundary)
- first grey value in a sequence of grey values (1 byte)
- minimal and maximal depth of the Huffman-code binary tree (2 bytes)
- binary string for Huffman-code binary tree (1 bit for each node between minimal and maximal depth, filled up with zeros to the next byte boundary)
- Huffman dictionary (less than 256 bytes)
- sequence of Huffman-coded grey values

We have also considered arithmetic coding [59] as an alternative to Huffman coding that adapts better to the distribution of grey values. In our experiments, however, no better compression of the grey value sequence was achieved in this way: For the image sizes used here, reductions in the length of the encoded sequence itself are compensated by the increased size of the coding dictionary.

## 3.3 Preprocessing by Requantisation

To further enhance BTTC, we introduced a modification that has not been exploited in the original paper by Distasi et al. [24]: We preprocessed the image by a (lossy) requantisation step that reduced the number of grey values from 256 to 64. As this shortens the grey value codes, it allows to retain more interpolation pixels at a given compression rate. The resulting gain in restoration quality exceeds the loss caused by the higher quantisation error.

## 3.4 Decoding by Linear Interpolation

Decompression takes place in two steps. In the first step, the vertex mask is recovered from the binary tree representation, and the stored grey values are placed at the appropriate pixel positions to give the sparse image. To recover the vertex mask, the tree is generated in the same order as it was stored. Along with generating nodes, vertex positions are calculated and marked in the vertex mask. The second step consists of the interpolation of the image, where the vertex mask becomes the interpolation mask. In the BTTC scheme of Distasi et al. [24], linear interpolation within each triangle is used.

In the sequel we will denote the entire coding and decoding technique of Distasi et al. by **Q64+BTTC(L)-L**. This nomenclature characterises a method with requantisation preprocessing to 64 grey levels, followed by B-tree triangular encoding where linear interpolation is incorporated, and decoded using linear interpolation.

# 4 EED-Based Coding and Decoding

Having seen in the previous section how images can be coded efficiently via binary trees, we are in a position to use these binary trees as backbone for our EED-based codec. While the format of the coded image remains, we introduce several amendments step by step. Each of them improves the quality of the restored image at a given compression rate.

## 4.1 Decoding with EED

Since we have observed that EED performs favourably as a scattered data interpolant, it is natural to renounce the linear interpolation step for decoding in the Q64+BTTC(L)-L method, and apply EED to the interpolation mask instead. We abbreviate this codec by **Q64+BTTC(L)-EED**. Our workshop paper [32] was based on this method.

Note that in contrast to Q64+BTTC(L)-L, the Q64+BTTC(L)-EED method does not rely on the triangulation, only on its vertices as interpolation points. Moreover, this replacement of linear interpolation by EED is the only step that affects the decoding. All amendments that are described next take place in the coding phase: They optimise the sparsification step which selects the scattered interpolation points and their stored grey values.

## 4.2 Adaptive Error Threshold

One approach to improve the compression procedure addresses the choice of the threshold parameter $\varepsilon$. In the original BTTC procedure, this parameter is constant throughout the construction of the binary tree. Recall, however, that the decision whether or not to split a triangle is based on the maximum error encountered within that triangle. Since splitting a large triangle affects many more pixels than splitting a small one, it is expected to be more efficient in reducing the average error. This suggests a strategy in which a more restrictive, i.e., smaller, threshold $\varepsilon$ is used on the coarser levels of the binary tree.

Since the size of the triangles shrinks exponentially with the level index of our binary tree, an exponential scaling is chosen for the threshold, too: Starting from a small threshold $\varepsilon_0$ on the coarsest level, it is adapted by a constant factor $\alpha > 1$ per level (typically, in the range between 1.35 and 1.5), resulting in

$$\varepsilon_k = \alpha^k \varepsilon_0 \qquad (16)$$

in level $k$ of the binary tree. We denote this algorithm with adaptive threshold by **Q64+BTTC(L,AT)-EED**.

## 4.3 Coding with EED

In Subsection 4.1 we have replaced the linear interpolation within each triangle in the *decoding* step by EED-based interpolation. This already leads to a considerable improvement in restoration quality. At the same time, a mismatch between coding and decoding was introduced in this way, since we retained the linear interpolation in the sparsification procedure.

As a remedy, diffusion-based interpolation can be used in the sparsification step, too. To decide whether to split a triangle, the current sparse image is completed by EED interpolation and compared to the original image. If the error of some pixel within the triangle in question exceeds the threshold pertaining to the current refinement level, the triangle is split.
In contrast to the linear interpolation used in Section 3, diffusion interpolation is not localised within each triangle: The changes in the interpolation result that occur when inserting a new vertex are not limited to the triangle that has been split, or even adjacent triangles. After updating the tree, one has therefore always to recompute the interpolation for the entire image, not just for a triangle. For best coding quality, recomputing is done after each insertion of a node. Alternatively, all decisions on the splitting of triangles of one and the same size can be made simultaneously, thereby restricting recomputations to one interpolation per triangulation level. This leads to a faster compression at the cost of a slight loss in image quality. In the current stage we are more interested in optimal quality than in fast algorithms. Thus we perform a recomputation for each inserted vertex. This EED-based compression method is called **Q64+BTTC(EED,AT)-EED**.

## 4.4 Biasing Interpolation Values

A close look at the PDE interpolation results in Figure 1 reveals that interpolation points that are local extrema tend to stand out in the interpolated image as sharp peaks. This effect is most striking in the linear diffusion result but can also be observed in less pronounced form in the nonlinear diffusion interpolation. In the case of linear interpolation, it can be related to the decay behaviour of radial basis functions that underly the steady state of the diffusion interpolation.
As a consequence, it is not always appropriate to store the exact grey values of the interpolation pixels extracted from the original image: While the reconstruction error in the interpolation pixel itself is minimised in this way, surrounding pixels can be considerably biased. For example, if the interpolation pixel is a local maximum, grey values in its neighbourhood will be systematically underestimated. Storing, instead, a slightly larger grey value for the interpolation pixel introduces an error in this particular pixel while reducing the error in its neighbourhood; see Figure 2. As the latter involves multiple pixels, the average absolute error is reduced.
A caveat in exploiting this idea is that it is difficult to estimate the influence zone of a given pixel in the interpolation of a scattered data set. Our realisation is based on the conservative assumption of a small influence zone given by a $3 \times 3$ neighbourhood. After determining the interpolation mask,

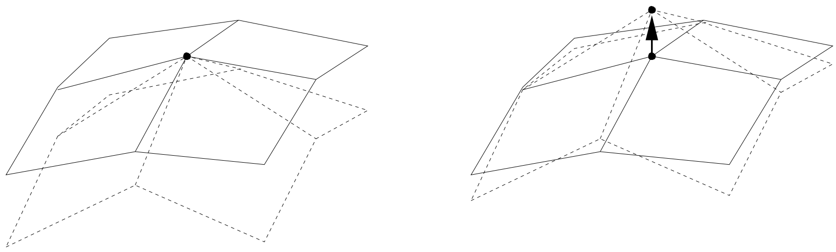

Figure 2: Biasing of an interpolation value. Vertical direction expresses grey values. An interpolation pixel (dot in the middle) with its $3 \times 3$ neighbourhood in the original image (joined by solid lines) and in the interpolated image (dashed lines). **Left:** Without biasing. **Right:** With biasing.

the interpolated image $u$ using exact grey values in all interpolation pixels is compared to the original image $v$. For each pixel $(i, j)$ of the interpolation mask we compute the average error

$$\tilde{e}_{i,j} := \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} (v_{k,l} - u_{k,l}) \tag{17}$$

and use it to correct the stored grey value for pixel $(i, j)$: Instead of $u_{i,j}$ we store the value

$$\tilde{u}_{i,j} := u_{i,j} + \tilde{e}_{i,j} \ . \tag{18}$$

Although a more precise estimation of the influence zones of interpolation pixels could most probably further improve the quality of restored images, already the simple procedure described here offers a measurable reduction of the reconstruction error. This version of the algorithm with biased interpolation is named **Q64+BTTC(EED,AT)+B-EED**.

## 4.5 Postprocessing by Requantisation

Finally, we revisit the requantisation procedure described in Subsection 3.3. By deferring the requantisation until the actual coding step, we gain a more precise error measurement during the sparsification procedure. On the other hand, the interpolated images in the sparsification procedure are computed based on grey values different from those that are eventually stored, which may deteriorate restoration results. The question which of these two settings, quantisation as preprocessing or postprocessing, is superior, is therefore difficult to answer theoretically.

Experiments, however, indicate that postprocessing is preferable. This is particularly true if it is applied in conjunction with the biasing of interpolation

values as described in the previous subsection: Bias corrections computed there may often be less than one quantisation level. Only when the quantisation acts as postprocessing, small adjustments can therefore optimally be accounted for in the quantised data set. Our best postprocessing results have been achieved with quantisation to 32 grey levels.
We will refer to the compression algorithm with post-quantisation to 32 levels by **BTTC(EED,AT)+B-Q32-EED**. Note the shifted order of the Q label. This is our most advanced algorithm that incorporates all amendments. For the sake of brevity, this EED-based codec is also abbreviated by **EEDC** when being compared with other algorithms such as JPEG or JPEG2000.

# 5 Experiments on Compression

Let us now investigate the effects of our EED-based interpolation in the context of image coding. In all experiments in this section, the parameters have been optimised in order to give the best compression quality.

Figures 3 and 4 show a test image and its compressed versions using Q64+ BTTC(L)-EED, our BTTC-method with EED decoding. We have chosen the threshold parameter $\varepsilon$ such that compressions of 0.8, 0.4 and 0.2 bits per pixel (bpp) are achieved. Compared to the standard coding that uses 1 byte per pixel, this comes down to compression ratios of 10:1, 20:1 and 40:1. In Figure 4, we display both the coded pixels with their corresponding grey values, and the result after scattered data interpolation with EED. While the image quality deteriorates with increasing compression, we observe that even at high compression rates, still fairly realistic results are possible. Therefore we carry out our further comparisons at the high compression rate of 40:1 (or equivalently 0.2 bpp) where the differences between the original and the compressed images are well visible.
In Figure 5, we compare Q64+BTTC(L)-EED with the original Distasi method Q64+BTTC(L)-L and the widespread JPEG compression. We show images reconstructed after compression to 0.2 bpp by each method. We observe that JPEG coding suffers from severe block artifacts that result from the fact that the discrete cosine transform is computed within blocks of $8 \times 8$ pixels. The Q64+BTTC(L)-L method, on the other hand, creates a different type of artifacts where the underlying triangulation becomes visible. Since Q64+BTTC(L)-EED only uses the interpolation points from the Q64+BTTC(L)-L method, but not the corresponding triangulation, it is clear that this method cannot suffer from such a shortcoming. Where insufficient data are available, its interpolation tends to be on the smoother side. As a

Figure 3: Test image *trui*, $257 \times 257$ pixels.

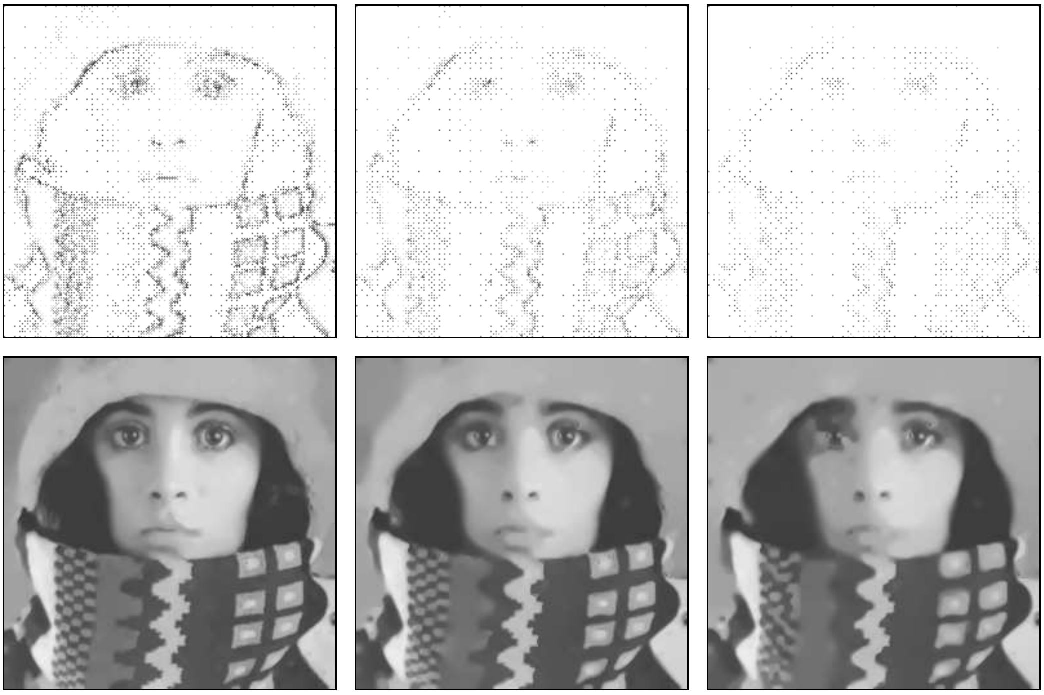

Figure 4: **First row, left to right:** Adaptive sparsification of *trui*, using BTTC with compression to 0.8 bpp, 0.4 bpp, 0.2 bpp. **Second row, left to right:** Corresponding interpolation (Q64+BTTC(L)-EED).

consequence, already this most basic one of our PDE-based algorithms gives results that are visually superior to the other two methods.

This visual impression is also confirmed by the quantitative measurements in Table 2, where the average absolute error is listed. We see that at the

Figure 5: Comparison at high compression rates (0.2 bpp) for the test image *trui*. **Left:** JPEG. **Middle:** Q64+BTTC(L)-L. **Right:** Q64+BTTC(L)-EED.

compression rate 40:1, JPEG performs worst, Q64+BTTC(L)-L is in the midfield, and Q64+BTTC(L)-EED gives the best results.

Table 2: Comparison of the average absolute error (AAE) for the *trui* image and compression to 0.2 bpp.

| codec | AAE |
|---|---|
| JPEG | 11.25 |
| Q64+BTTC(L)-L | 8.63 |
| Q64+BTTC(L)-EED | 8.45 |

In Figure 6 we demonstrate how the quality of reconstructed images is successively improved by introducing the adaptive threshold, diffusion-based sparsification, biasing and post-quantisation. Each step constitutes an improvement over its predecessor. In compression methods using EED in the coding step, interpolation is recomputed once per node to achieve optimal quality. With fast compression (a single EED interpolation per level) the AAE values deteriorate by approximately 0.1 to 0.2.
A comparison in terms of their average absolute error in Table 3 further supports these observations. In this table we include also the error measure for JPEG2000. For the BTTC-based methods, we give also the numbers of pixels in the sparse images. From these numbers it can be read off firstly that all algorithmic modifications improve the quality of selected pixels, while their number remains in the same range between 1500 and 1550 pixels. Secondly, it is confirmed that a reduced number of quantisation levels allows to store substantially more pixels.

Figure 6: Comparison of BTTC-based compression methods at a compression rate of 0.2 bpp. **First row, left to right:** EED-based decoding (Q64+BTTC(L)-EED), with adaptive threshold (Q64+BTTC(L,AT)-EED), with EED coding (Q64+BTTC(EED,AT)-EED). **Second row, left to right:** with biasing (Q64+BTTC(EED,AT)+B-EED), with post-quantisation to 64 grey levels (BTTC(EED,AT)+B+Q64-EED), with post-quantisation to 32 grey levels instead (BTTC(EED,AT)+B+Q32-EED).

We extend this comparison in Table 4 for both pre- and postprocessing quantisation, including also the case without quantisation (256 grey levels). It can be seen that the number of pixels increases with coarser quantisation. For very small numbers of grey levels, however, the growing quantisation error dominates over the AAE improvements by having more pixels, making 32 the preferable choice. Post-quantisation offers an advantage over pre-quantisation for medium quantisation levels that disappears when reducing the number of quantisation levels to 16.

Our most optimised EED-based codec is given by BTTC(EED,AT)+B+Q32-EED (from now on simply called EEDC). It achieves an average absolute error of 4.99. This is far better than JPEG with an error of 11.25 and very close to the 4.86 value for JPEG2000 which marks the state of the art in image compression. For a visual comparison, we juxtapose this algorithm

Table 3: Comparison of absolute errors for different methods at 0.2 bpp. For BTTC-based methods, the number of pixels in the sparse image is stated in the last column.

| codec | AAE | #pixels |
|---|---|---|
| Q64-BTTC(L)-EED | 8.45 | 1543 |
| Q64-BTTC(L,AT)-EED | 5.98 | 1517 |
| Q64-BTTC(EED,AT)-EED | 5.55 | 1542 |
| Q64-BTTC(EED,AT)+B-EED | 5.32 | 1530 |
| BTTC(EED,AT)+B+Q64-EED | 5.27 | 1527 |
| BTTC(EED,AT)+B+Q32-EED | 4.99 | 1769 |
| JPEG2000 | 4.86 | |

Table 4: Comparison of absolute errors and numbers of stored pixels for different quantisation levels. We juxtapose pre-quantisation (Q**+BTTC(EED,AT)+B-EED) and post-quanitsation (BTTC(EED,AT)+B+Q**-EED).

| #quantisation levels | pre-quantisation | | post-quantisation | |
|---|---|---|---|---|
| | AAE | #pixels | AAE | #pixels |
| 256 | 6.42 | 1123 | 6.42 | 1123 |
| 64 | 5.32 | 1530 | 5.27 | 1527 |
| 32 | 5.04 | 1770 | 4.99 | 1769 |
| 16 | 5.33 | 2099 | 5.34 | 2054 |

Figure 7: **Left:** Original image *trui*. **Middle:** Reconstructed after compression to 0.2 bpp by JPEG2000 (AAE = 4.86). **Right:** Reconstructed after compression to 0.2 bpp by EEDC (AAE = 4.99).

and JPEG2000 in Figure 7. It can be seen that the overall visual quality of both methods is very similar. Each algorithm represents some details better. The result of the EEDC algorithm appears slightly smoother and is free from the faint block artifacts that occur even in JPEG2000. On the other hand, it tends slightly more towards cartoon-like smoothed structures.

In Figure 8, we show the relation between EEDC, JPEG, and JPEG2000 over a range of different compression ratios. At the lowest compression ratio 5:1, the differences between all methods are marginal. While JPEG deteriorates rapidly at higher compression ratios, both JPEG2000 and EEDC show a moderate increase in the reconstruction error.

Finally, a comparison of EEDC with JPEG and JPEG2000 on a variety of popular compression test images is shown in Figure 9 and Table 5. It demonstrates that in terms of reconstruction quality at high compression ratio, PDE-based compression consistently ranks between JPEG and JPEG2000, and is always closer to the latter. For images with a lot of detail and textures, like *barbara* or *boats*, the tendency to smoothing becomes more striking. In these cases, however, even JPEG2000 displays a noticeable loss of detail.

# 6 Summary and Conclusions

While contemporary image compression is dominated by transform-based methods that rely on the discrete cosine transform or wavelet decompositions, we have shown that PDEs have the potential to become a serious alternative. To this end, we have driven PDE-based inpainting ideas to the extreme by storing only a sparse set of all pixels and interpolating the missing

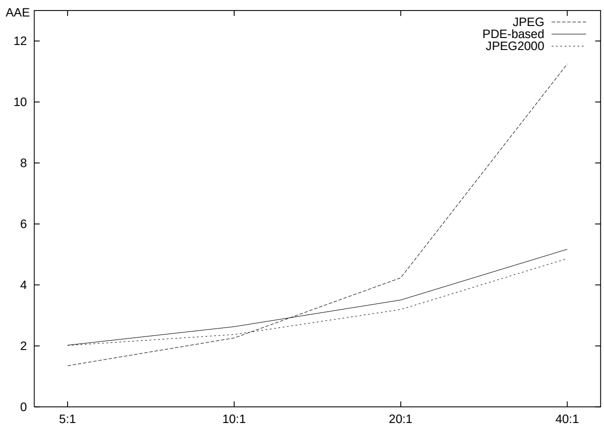


Figure 8: Comparison of EEDC with JPEG and JPEG2000 over a range of compression ratios from 5:1 (1.6 bits per pixel) to 40:1 (0.2 bpp) for the test image *trui*.

Table 5: Comparison of absolute average errors for different images and compression methods at 0.2 bpp, as shown in Figure 9.

| Image | JPEG | EEDC | JPEG2000 |
|---|---|---|---|
| *lena* | 13.61 | 8.72 | 7.20 |
| *cameraman* | 13.75 | 9.38 | 8.07 |
| *peppers* | 12.19 | 7.53 | 6.59 |
| *barbara* | 15.47 | 12.22 | 10.06 |
| *boats* | 14.68 | 11.82 | 9.28 |

Figure 9: Comparison of compression methods at 0.2 bpp for different test images (rescaled to $257 \times 257$ pixels). **Rows, top to bottom:** *lena*, *cameraman*, *peppers*, *barbara*, *boats*. **Columns, left to right:** Original image, reconstructed images after compression by JPEG, by EEDC, and by JEPG2000.

data by edge-enhancing anisotropic diffusion. The sparse point set has been constructed by a B-tree triangular coding with several improvements such as the adaptation of the threshold parameter, diffusion-based point selection, and a specific quantisation strategy. The fact that with this relatively moderate degree of sophistication, our EED-based codec clearly outperforms the JPEG standard at high compression rates and even comes close to the quality of the highly optimised JPEG2000 is very encouraging.

With respect to the specific method in our paper, there is certainly room for further optimisation, for instance by incorporating ideas from rate distortion theory. In order to establish PDE-based methods as a more general alternative to existing paradigms, additional research is underway, e.g. with respect to the theoretical foundations of diffusion-based interpolation [9], alternative sparsification strategies [21, 9] including feature-based ones [76], compact representation of textured regions, highly efficient numerical algorithms [42], as well as generalisations to vector- and tensor-valued images [69], image sequences [42], and surface data [5]. More details will be reported in forthcoming publications.

# Acknowledgements

Our research is partly funded by the *International Max-Planck Research School (IMPRS)*. This is gratefully acknowledged. Joachim Weickert also thanks Vicent Caselles for interesting discussions on EED-based interpolation during a stay at the University Pompeu Fabra, Barcelona.

# References

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

[2] 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, October 2005.

[3] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, volume 147 of *Applied Mathematical Sciences*. Springer, New York, 2002.

[4] 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.

[5] E. Bae. New PDE-based methods for surface and image reconstruction. Master's thesis, Dept. of Mathematics, University of Bergen, Norway, 2007.

[6] R. Bajcsy and S. Kovačič. Multiresolution elastic matching. *Computer Vision, Graphics and Image Processing*, 46(1):1–21, April 1989.

[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, Montova, Italy, September 2003. IEEE Computer Society Press.

[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, October 2004.

[9] Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert. How to choose interpolation data in images, 2008. In preparation.

[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. M. Bruckstein. On image extrapolation. Technical Report CIS9316, Computer Science Department, Technion, Haifa, Israel, April 1993.

[13] 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, February 2006.

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

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

[16] 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.

[17] T. F. Chan and J. Shen. *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*. SIAM, Philadelphia, 2005.

[18] T. F. Chan and H. M. Zhou. Feature preserving lossy image compression using nonlinear PDE's. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, volume 3461 of *Proceedings of SPIE*, pages 316–327. SPIE Press, Bellingham, 1998.

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

[20] 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.

[21] H. Dell. Seed points in PDE-driven interpolation. Bachelor's Thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany, 2006.

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

[23] 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 Technolgy, Cambridge, MA, U.S.A., November 1996.

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

[25] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *RAIRO Mathematical Models and Methods in the Applied Sciences*, 10:5–12, 1976.

[26] S. Durand and M. Nikolova. Restoration of wavelet coefficients by minimizing a specially designed objective function. In O. Faugeras and N. Paragios, editors, *Proc. Second IEEE Workshop on Geometric and*

*Level Set Methods in Computer Vision*, Nice, France, October 2003. INRIA.

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

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

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

[30] G. E. Ford, R. R. Estes, and H. Chen. Scale-space analysis for image sampling and interpolation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 165–168, San Francisco, CA, March 1992.

[31] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38:181–200, 1982.

[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] A. Gothandaraman, R. Whitaker, and J. Gregor. Total variation for the removal of blocking effects in DCT based encoding. In *Proc. 2001 IEEE International Conference on Image Processing*, volume 2, pages 455–458, Thessaloniki, Greece, October 2001.

[34] H. Grossauer and O. Scherzer. Using the complex Ginzburg–Landau equation for digital impainting in 2D and 3D. In L. D. Griffin and M. Lillholm, editors, *Scale-Space Methods in Computer Vision*, volume 2695 of *Lecture Notes in Computer Science*, pages 225–236. Springer, Berlin, 2003.

[35] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

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

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

[38] T. Iijima. Basic theory on normalization of pattern (in case of typical one-dimensional pattern). *Bulletin of the Electrotechnical Laboratory*, 26:368–388, 1962. In Japanese.

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

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

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

[42] 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.

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

[44] T. Lehmann, C. Gönner, and K. Spitzer. Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, November 1999.

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

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

[47] 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.

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

[49] R. March. Computation of stereo disparity using regularization. *Pattern Recognition Letters*, 8:181–187, October 1988.

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

[51] E. Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, March 2002.

[52] J. Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, Oxford, 2004.

[53] P. Mrázek. *Nonlinear Diffusion for Image Filtering and Monotonicity Enhancement*. PhD thesis, Czech Technical University, Prague, Czech Republic, June 2001.

[54] H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:565–593, 1986.

[55] G. M. Nielson and J. Tvedt. Comparing methods of interpolation for scattered volumetric data. In D. F. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics: Aspects of Visualization*, pages 67–86. Springer, New York, 1994.

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

[57] 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.

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

[59] J. Rissanen and G. G. Langdon Jr. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162, 1979.

[60] 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.

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

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

[63] 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.

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

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

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

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

[68] J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner, Stuttgart, 1998.

[69] 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.

[70] 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 5266 of *Proceedings of SPIE*. SPIE Press, Bellingham, 2007.

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

[72] S. Yang and Y.-H. Hu. Coding artifact removal using biased anisotropic diffusion. In *Proc. 1997 IEEE International Conference on Image Processing*, volume 2, pages 346–349, Santa Barbara, CA, October 1997.

[73] S. Yao, W. Lin, Z. Lu, E. P. Ong, and X. Yang. Adaptive nonlinear diffusion processes for ringing artifacts removal on JPEG 2000 images. In *Proc. 2004 IEEE International Conference on Multimedia and Expo*, pages 691–694, Taipei, Taiwan, June 2004.

[74] N. Yokoya. Surface reconstruction directly from binocular stereo images by multiscale-multistage regularization. In *Proc. Eleventh International Conference on Pattern Recognition*, volume 1, pages 642–646, The Hague, The Netherlands, August 1992.

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

[76] H. Zimmer. PDE-based image compression using corner information. Master's thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany, 2007.