Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 354

Beyond Pure Quality: Progressive Modes, Region of Interest Coding, and Real Time Video Decoding for PDE-based Image Compression

Pascal Peter, Christian Schmaltz, Nicolas Mach,
Markus Mainberger and Joachim Weickert

Saarbrücken 2015

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 354
submitted: January 30, 2015

Beyond Pure Quality: Progressive Modes, Region of Interest Coding, and Real Time Video Decoding for PDE-based Image Compression

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

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

Nicolas Mach
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Campus, E1.7
Saarland University, 66041 Saarbrücken, Germany
mach@mia.uni-saarland.de

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

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

Compared to well-established transform-based image compression methods such as JPEG and JPEG 2000, approaches based on partial-differential equations (PDEs) are still in a proof-of-concept stage. Nevertheless, they have already surpassed JPEG 2000 quality-wise for medium to high compression rates for certain classes of images. This particularly holds for R-EED, a codec employing edge-enhancing anisotropic diffusion (EED) and rectangular subdivision. However, today's requirements for practically viable compression algorithms go beyond pure compression performance. Codecs must also fulfill the individual feature requirements of specific applications such as internet media or medical imaging. In this paper we propose three such features for the R-EED codec. By reordering grey values and exploiting the subdivision scheme, we incorporate a progressive mode into R-EED. In experiments we can verify that this extension outperforms JPEG and JPEG 2000. Additionally, we show that the rectangular subdivision is well-suited for the incorporation of region of interest (ROI) coding. Our ROI-extensions allows to store image parts with different quality depending on their importance. Finally, to demonstrate that R-EED-based decoding can be performed efficiently, we propose a real-time video player that uses R-EED. All of these extensions are compatible with each other and can be used simultaneously.

1 Introduction

Lossy image compression aims at reducing the file size of an image while degrading the image quality as little as possible. One of the most prominent and widely used lossy image compression algorithms today is JPEG [1], which is based on the discrete cosine transform. Its successor JPEG 2000 [2], which uses Cohen-Daubechies-Feauveau wavelets, performs noticeably better, but is not yet widely supported.

Recently, these traditional transformation-based approaches have been challenged by novel compression methods that rely on an entirely different concept: Algorithms that rely on partial differential equations (*PDE*s) only store a subset of all image points and reconstruct the remainder of the image with PDE-based interpolation.

While there is a long history on research on feature-based image representations where homogeneous diffusion fills in missing information (see e.g. [3, 4, 5, 6, 7]), most of the early works do not consider applications in image compression. Only recently it has been shown that edge- or segment-based homogeneous diffusion approaches can beat JPEG 2000 for cartoon-like im-

ages [8], as well as for depth maps [9, 10, 11]. Apart from these linear, feature-based approaches, there are some attempts to use nonlinear PDEs for image compression. Chan and Zhou [12] propose a variational approach with total variation regularisation to minimise oscillations in wavelet decompositions. Work by Solé *et al.* [13] evaluates different PDEs for compression of digital elevation maps. Liu *et al.* integrate inpainting into existing codecs [14]. The image compression algorithm of Galić *et al.* [15] uses points located on an adaptive triangulation and stores these locations as a binary tree. This method reconstructs the remainder of the image with edge-enhancing anisotropic diffusion (EED) [16]. Even with the improvements introduced in [17], the performance of this codec only lies between that of JPEG and JPEG 2000 for medium to high compression ratios. The R-EED approach of Schmaltz et al. [18] builds upon these ideas. By introducing a number of novel concepts, e.g. using rectangular subdivisions instead of triangular ones, the obtained image compression codec yields better results. They can even surpass those of JPEG 2000 for most compression ratios in images that are not dominated by texture. Schmaltz *et al.* demonstrated in [19] how one can further improve the compression quality of the R-EED algorithm.
However, a good reconstruction at high compression ratios alone is not sufficient, since there are more requirements for an image compression algorithm in practice. Depending on the field in which the codec is applied, such requirements differ. For example in medical imaging, a high reconstruction quality in diagnostically relevant regions is essential, while other regions can contain more errors. In web-based applications, codecs have to be able to deal with partially transferred files to reduce load times, and in time critical applications such as video playback, real-time capabilities are important. In this paper, we address such additional requirements for R-EED. In particular, we focus on progressive modes, region of interest coding, and real-time video decoding.

1.1 Related Work

Progressive mode. The progressive mode, which is also referred to as embedded bitstreams or signal-to-noise ratio scalability, allows to generate a coarse preview from the beginning of a complete data stream only. This is especially advisable for applications in which only a limited bandwidth is available, e.g. when browsing a database of large images.
Progressive modes are readily available in most standard image compression codecs. Lossless image compression algorithms typically use interleaving or interlacing, which only changes the order in which pixels are stored. The well-known GIF file format [20], for example, divides the image into stripes

with a height of eight pixels each, and stores the resulting lines in each stripe during multiple passes. PNG [21] employs a similar idea known as *Adam7*, which samples in both directions: It subdivides the image into 8×8 blocks and stores them in a total of seven passes.
The lossy mode of the JPEG standard provides several optional progressive modes: The progressive spectral selection reorders the transmission of the DC and AC components such that all low-frequent coefficients are transmitted first. Alternatively, one can successively approximate the stored coefficients by first storing the upper bits of the coefficients, while the lower bits follow later. It is possible, albeit unusual, to use both approaches at the same time. From a purely qualitative perspective, the so-called hierarchical mode provides typically the best results at low bit rates. This mode first stores a downsampled version of the image. The upsampled, bi-linearly interpolated version of this smaller image acts as a predictor for the next resolution. However, the total file size can increase up to one third with this mode [1].
In JPEG 2000, images are always stored in progressive mode due to the the *Embedded Block Coding with Optimal Truncation* (*EBCOT*) scheme. EBCOT encodes the bit-planes in three passes. In which pass a bit gets encoded depends on which coefficients have significant neighbours or are significant themselves.
So far, none of the PDE-based image compression methods discussed above include progressive modes. Note that standard progressive modes of existing image compression algorithms are not directly applicable to PDE-based methods, since data is only available at irregularly placed points.

Region of interest coding. There is no direct support for region of interest coding in JPEG. Nevertheless, a simple standard compliant idea proposed is to set additional DC coefficients to zero outside the region of interest [1].
JPEG 2000 contains many methods for region of interest coding, namely a general scaling based approach, a bitplane-by-bitplane shift method (BbB-Shift), a max-shift method, a partial significant bitplane shift method (PSBShift) and a ROI coding through component priority (ROITCOP). Due to the large amount of approaches, we refer to [22, 23] for details.
To the best of our knowledge, no PDE-based codec with ROI coding capabilities has been presented so far.

Real-time video decoding. Almost the whole existing body of work on PDE-based compression focuses on still image compression. However, there are some notable exceptions from the rule. Gao [24] used diffusion-based inpainting to compress optic flow fields for motion compensation in video coding. However, the actual compression of video data in this approach re-

lies on block-coding with the discrete cosine transform. Therefore, the codec is still mainly transform-based. So far, real-time video decoding with fully PDE-based algorithms has only been implemented by Köstler *et al.* [25] and Baum [26]. Note that the work of Köstler *et al.* is based on linear diffusion which is computationally much less demanding than the high quality reconstructions of nonlinear anisotropic methods such as R-EED. Although the approach of Baum allows both linear and nonlinear anisotropic inpainting, it only achieves real-time decompression at the price of a noticeably degraded compression quality.

1.2 Goals

Consequently, the goal of this paper is threefold: First we give a detailed description how progressive modes can be incorporated into a current PDE-based image compression algorithm. This first part refines and expands work previously presented in [27]. Secondly, we extend our algorithm such that it includes a region of interest coding. This allows us to specify which parts of the image are important and should be saved with a higher precision. Finally, we introduce a PDE-based video decoder that allows real-time playback with nonlinear anisotropic diffusion inpainting.
Since R-EED is currently the most advanced PDE-based general purpose codec that can surpass JPEG 2000 (see [18]), we use it as the basis of our approach. However, we have modified R-EED slightly to further enhance the performance of the algorithm independently of the novel features. Additionally, all three extensions are designed to be compatible, i.e. progressive mode and region of interest coding can both be applied simultaneously and are also applicable to video data.

1.3 Paper Structure

First, we provide a brief overview on PDE-based compression and the baseline codec R-EED in Section 2. Section 3 contains a detailed discussion of the progressive mode for R-EED as well as comparisons to JPEG and JPEG 2000. We introduce the second extension of R-EED, region of interest coding, in Section 4 and demonstrate its capabilities experimentally. In Section 5, we propose a real-time video-player based on R-EED compression and compare its performance to reference results obtained from classical solvers for PDE-based inpainting. Finally, we conclude the paper with a summary and an outlook in Section 6.

2 The Baseline Image Compression Codec

The R-EED codec [19] only stores the image values at a few selected locations, and employs PDE-based interpolation to reconstruct the missing data when loading the image. Thus, we first explain the ideas behind inpainting with PDEs. A description of the complete R-EED algorithm follows in Section 2.2.

2.1 PDE-Based Interpolation

Let $f : \Omega \to \mathbb{R}$ denote a grey value image with a rectangular image domain $\Omega \subset \mathbb{R}^2$. In PDE-based image interpolation, the grey values are only known at the locations of the *interpolation mask* $K \subset \Omega$ of the complete image f. The additional assumption that the pixels in $\Omega \setminus K$ are regular in some sense allows to reconstruct the unknown part of the image from the known grey values in K. For example, one might assume that the reconstructed image u is piecewise smooth in $\Omega \setminus K$. To fulfil both conditions, we solve the partial differential equation (PDE)

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

for u with reflecting (i.e. homogeneous Neumann) boundary conditions. Thereby, c_K is the *characteristic function* of K, i.e. a function that has value 1 at the specified data set K, and 0 elsewhere. The differential operator L ensures the requested smoothness properties. Since this PDE does not change the known points in K, we can rewrite it as

$$Lu = 0 \tag{2}$$

on $\Omega \setminus K$, with Dirichlet boundary conditions on K, and homogeneous Neumann boundary conditions on the boundary of Ω. One can derive Equation (1) also as a generalisation of spline interpolation and variational regularisation, as is explained in [28].
To solve Equation (2), we introduce an artificial time parameter t and compute the steady state of the evolution equation $\partial_t u = Lu$ with a finite difference approximation. For the image compression framework used in this paper, we employ the same inpainting operator as proposed in [17, 19], namely *edge-enhancing anisotropic diffusion* (*EED*) [16]:

$$Lu = \operatorname{div}(\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)\boldsymbol{\nabla} u). \tag{3}$$

In this equation, u_σ denotes the image u after smoothing with a 2-D Gaussian K_σ with standard deviation σ. The diffusion tensor $\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)$ is a symmetric,

positive definite 2×2 matrix. Its eigenvectors are $\boldsymbol{\nabla} u_\sigma$ and $\boldsymbol{\nabla} u_\sigma^\perp$, which are oriented across image edges, and along them, respectively. The corresponding eigenvalues are given by

$$\mu_1 = \frac{1}{\sqrt{1 + \frac{|\boldsymbol{\nabla} u_\sigma|^2}{\lambda^2}}}, \quad \mu_2 = 1. \tag{4}$$

Consequently, we obtain a smooth interpolation along edges and a discontinuity-preserving interpolation across edges. A careful adaptation of the contrast parameter λ to the characteristics of the reconstructed image can improve the inpainting results.

2.2 The R-EED Codec

The core idea behind the R-EED codec is to store only a small inpainting mask K and the corresponding grey values. Therefore, a central question is how to choose the set K for a given image. On the one hand, the set K should allow a good reconstruction quality. On the other hand, it should be possible to encode the set K and the corresponding grey values efficiently. Thus, R-EED does not try to find optimal point positions, as is done in [8], for example. Instead, a recursive rectangular subdivision scheme restricts the points in K. In each rectangular region obtained by subdivision, R-EED only stores the centre and the four corner pixels. Starting from the complete image, the codec first simulates the inpainting step with these known points. By means of the mean square error (MSE), this local reconstruction is compared against the original. If the MSE is smaller than a threshold T the local reconstruction is sufficiently accurate and the region is not divided further. In order to adapt this threshold to the varying size of the sub-images, it is defined as $T = a\ell^d$. Here, a is a global threshold parameter and ℓ a level adaption factor that is rescaled exponentially by the recursion depth d. Otherwise, R-EED splits the current sub-image along the middle of its longer side, and processes both image parts recursively as described above.

The splitting steps described in the last paragraph determine all points in the inpainting mask K: For each rectangle considered, K contains its centre and corners. Thus, a binary decision tree that represents which rectangles were split fully characterises K and allows to encode point positions efficiently. In order to further reduce the amount of disk space, R-EED first stores two depth limits: the largest depth until which all sub-images are split and the smallest depth at which no sub-image is split any more. Thus, it is not necessary to save the tree structure outside these limits. In between the

limits, a single bit marks if the node either has no child nodes at all or exactly two child nodes. Consequently, the smaller the distance between the two depth limits, the less bits are necessary to store the tree and therefore the inpainting mask. Limiting this range by introducing depth constraints in addition to the threshold T can thus help to reduce the storage cost for the tree. This additional bit budget is invested into additional mask points and thus improves the overall reconstruction quality.

In order to reconstruct the image, one also needs the grey values of the pixels in the mask K. R-EED first quantises these pixel values and encodes them with a general purpose entropy coding scheme. As explained in detail in [18, 19], the encoder PAQ [29] yields the best results for all except very small files, for which adaptive arithmetic coding [30] works best. However, PAQ is very slow and memory consuming which makes it ill-suited for our application in real-time video display. Moreover, due to its high complexity, PAQ is difficult to adapt to our requirements. Due to these reasons, we will only present results generated with either Huffman coding [31] or arithmetic coding.

R-EED optimises the contrast parameter λ from Equation (4), and stores the quantised λ-value that yields the smallest overall reconstruction error. Since this requires only a single byte, it does not result in a large overhead. Additionally, it is possible to encode optimised brightness values instead of the actual brightness in the input image. Despite the fact that this leads to a higher error at the known pixel locations, the overall reconstruction quality can be improved this way. Here, we employed a straightforward approach that optimises the grey values one after each other in random order. More advanced approaches are explained in detail in [8].

Especially when using high compression ratios, it can occur that the reconstructed values are actually more accurate than the stored brightness values in the inpainting mask. This is caused by the quantisation and the optimisation steps explained above. Following the ideas presented in [32] and [18], we thus perform additional steps after inpainting: First, we mark all points whose distance to a point in the inpainting mask K is smaller than a threshold as unknown, while the remaining reconstructed points are assumed to be known. Then, a second inpainting step is performed. The optimal threshold depends on the image and the compression ratio, and is thus stored in the file header. Fore more details about the different steps of the R-EED algorithm, we refer to [19].

(a) (b) (c)

Figure 1: **(a)+(b)** Input images used in the experiments. **(c)** Image reconstructed from 50% of the data at a compression rate of 40 : 1 without using the progressive mode proposed in this paper.

3 R-EED in Progressive Mode

3.1 Algorithm

The goal of our progressive mode is to provide a preliminary reconstruction of the compressed image when only a part of the compressed image has been transmitted so far. Thus, the brightness values are unknown for some mask points and only the known part of the inpaintig mask is used for reconstruction.

The original R-EED algorithm employs an easy and straightforward way to store the brightness values indicated by the inpainting mask, namely a rowwise storage from top to bottom and left to right. For progressive mode, this ordering is not very useful: A partially transmitted file only contains brightness values of points in the upper part of the image. The pixel values from the lower part are still missing due to their location at the end of the compressed file. Figure 1(c) demonstrates that reconstructions from partial standard R-EED files are not satisfying.

Consequently, one of the core ideas of our progressive mode for R-EED is to change the order in which the algorithm stores the mask points. We save the brightness values in such a way that for any given percentage of the compressed file, known pixel values are distributed over the entire image domain. While this idea is straightforward, choosing the best order is nontrivial. Even if we knew in which order the points should be stored to obtain the best results, saving this order would be far too expensive in general.

As an example, consider a compressed file that stores only 720 grey values.

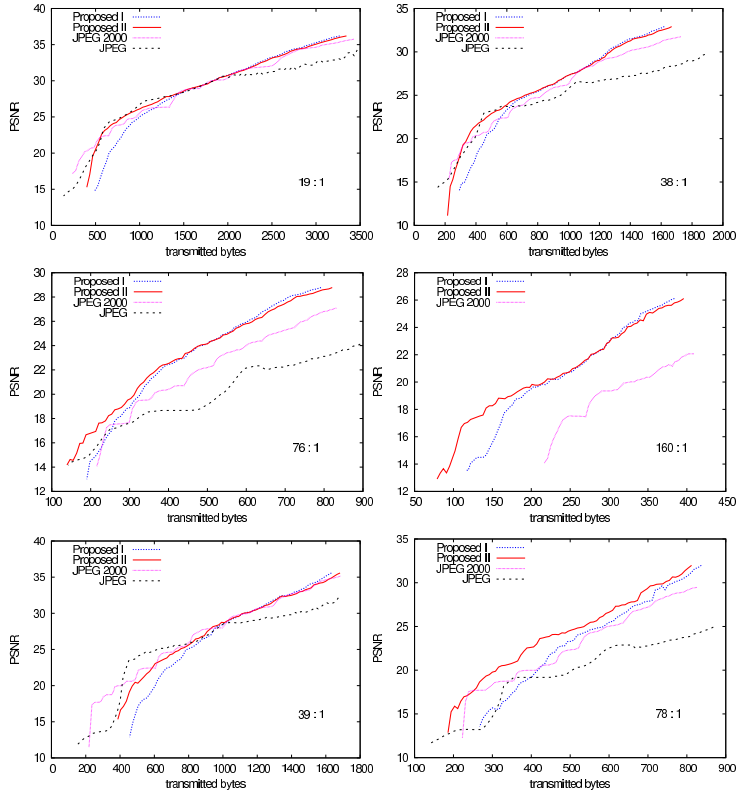


Figure 2: These graphs show the performance of JPEG, JPEG 2000, and of our progressive modes for the image *trui* (size: 256×256, see Figure 1(a)) and compression ratios around $19 : 1$ (top left), $38 : 1$ (top right), $76 : 1$ (middle left), and $160 : 1$ (middle right). JPEG is missing from the last of the four *trui* plots since it is unable to reach the requested compression ratio. The last row contains plots for the test image *walter* at the ratios $39 : 1$ and $78 : 1$.

Since there are 720! possible ways to order these points, on average, at least $\log_2(720!) > 5800$ bits $= 725$ bytes are necessary to store such an order. However, for an image such as *trui* from Figure 1, 720 points correspond to a compression ratio of $150 : 1$, or a file size of 437 bytes, respectively. That is, saving this order would increase the file size by 160%. Since the number of possible orders rises more rapidly than the exponential function in the number of points, the situation is even worse for smaller compression ratios. Instead of storing the best order in the compressed file, we therefore compute a good approximation from the information that is available for the decoder. This way, the file size of the compressed image does not increase, and we avoid expensive computations for finding the best order during the compression

stage. In particular, we exploit the fact that the subdivision scheme from the last section ensures that each image part contains the fraction of points that is necessary to obtain a consistent reconstruction quality.
For our progressive mode, we try to maintain these fractions no matter which amount of the compressed file is already known. Following this core idea, a region for that only a small percentage of its total known pixel values are already stored in the compressed file is called underrepresented. Consequently, during the sequential storage of the grey values, R-EED always chooses the next value from the most underrepresented image part. In the following, this approach is denoted by *R-EED-P1*.
More precisely, for each point to be saved, we traverse the tree starting from its root. In each node, we move to the child node which has a smaller percentage of already stored points. Once we arrive at a tree leaf – or a node for which all children have already stored all their mask points – we save the brightness of one unsaved mask point in the image part corresponding to this tree node. Since the image header contains the tree, no additional overhead is required to reproduce the same order during decompression. Before reconstructing a partially transmitted image, we remove mask points with hitherto unknown brightness values.
In addition to R-EED-P1, we have implemented and evaluated a number of other approaches to find good orders. Examples include a pseudo-random permutation, a level-wise storage of the points depending on the depth of their corresponding tree node, or the respective inverse of these orders. The idea behind using a reverse order is to reconstruct details early. However, the reconstruction of the coarse image features noticeably suffers in this approach if only small file parts are known. Slightly better results can be obtained this way once a large part of the file is known. Nevertheless, the significantly worse reconstruction for small parts of the compressed image render this approach clearly inferior to the approach R-EED-P1. However, to avoid possible confusions, we choose not to describe and compare against those inferior methods in detail here.
As long as only a very small part of the compressed image is known, the inpainting result is unsatisfying, since the information available is insufficient to reconstruct even basic image structures. Note that this problem is amplified by the overhead necessary to store the file header. The results of R-EED-P1 with 25% known data in Figure 3 illustrate this problem.

Our second progressive mode, *R-EED-P2*, improves the quality of reconstructions from small known parts of the compressed image. Its motivation stems from the observation that in R-EED, the number of stored points is closely related to the amount of quantisation. For increasing compression ratios,

both the number of points in the inpainting mask and the number of possible quantised values decrease in R-EED to guarantee the best compression results. Given only a small part of the compressed image, though, we know only very few points, while the quantisation levels are the same as before.
This imbalance is eliminated in our second approach by storing the brightness values in two parts. The first part allows only a coarse quantisation, while both parts together result in the same quantisation levels as without using a progressive mode. This permits to store more brightness values per byte, since less information is necessary to store a more coarsely quantised value. In addition, the separation of the brightness values also enables us to use two small Huffman trees instead of one large one. Since the second tree is not needed initially, we move out of the file header. Consequently, the total file header is noticeably smaller than before which makes the first approximation of the stored image available after a much smaller number of transmitted bytes. The graphs in Figure 2 illustrate this effect.
In the following we describe how R-EED-P2 splits quantised grey values with q quantisation levels. The first part consists of the integer values $[f_i/[\sqrt{q}]]$, where f_i is the i-th grey value and $[\cdot]$ denotes rounding to the closest integer. Afterwards, we store the remainder $r_i := f_i \mod [\sqrt{q}]$.
Note that it does not pay off to store all grey values in two parts. This would result in the same imbalance as if all points were given in a coarse quantisation. That is, once a certain amount of points is stored, it is possible to reconstruct the position of most image structures from these points. At this point, due to the very coarse quantisation of the known data, the reconstruction quality benefits more from increasing the accuracy of given grey values than from additional known data with low grey value accuracy.
Consequently, it pays off to save the missing remainders r_i next instead of storing even more coarsely quantised grey values. As the amount of points necessary to reconstruct the image structures varies with the image and the compression ratio, we use 5 bits in the file header to indicate how many percent of the points are stored in two parts.
In total our algorithm incorporates three passes: In the first step, we store the first Huffman tree and a part of the coarsely quantised brightness values. In the second pass, we first save the second Huffman tree, followed by the remainders r_i of the brightness values from the first step. Finally, our algorithm stores the grey values completely missing so far. Since storing a novel Huffman tree for those values would result in a noticeable overhead, we use the two known Huffman trees to store these values. That is, R-EED-P2 saves the coarse quantisation and the remainder consecutively for each point.

3.2 Experiments

We evaluate our proposed progressive modes on different images, varying compression rates and partial file sizes. As an error measure, we employ the well-known peak signal to noise ratio (PSNR). Figure 2 shows the resulting PSNRs for all intermediate file sizes of the images *trui* and *walter*. The examined compression ratios vary from 19 : 1 to 160 : 1. Figure 1 contains the uncompressed images. To allow comparisons, we also present the resulting error graphs for JPEG and JPEG 2000. The compressed JPEG and JPEG 2000 images were created using the Linux tool *convert* (Version ImageMagick 6.8.3-6 2013-03-04 Q16), while we used *Irfanview* to read partial JPEG 2000 files.

Figure 2 reveals several interesting facts. First of all, R-EED-P2 usually results in slightly larger files than R-EED-P1. This is due to the fact that entropy coding is less efficient when every grey value is stored in two passes. However, there are also cases in which it is the other way round, e.g. when compressing the image *walter* with a compression ratio of 78 : 1 in Figure 2. This happens when the space necessary to store two small entropy coding functions is smaller than the one necessary to store a large one. Depending on which of those two effects is stronger, the total file can be smaller or larger.

Even though the total file size is typically slightly larger when using R-EED-P2, this approach allows a much higher PSNR after a small amount of bytes have been transmitted. The first row of Figure 3 clearly demonstrates this effect. There are two reasons for this behaviour: On one hand, each grey value requires less bits due to the initially coarse quantisation which results in a larger amount of known grey values at the price of a lower accuracy. On the other hand, the overhead necessary to store the entropy coding scheme is smaller. Therefore, R-EED-P2 is also well-suited for high compression ratios where the amount of available data in progressive mode is severely reduced. As Figure 4 demonstrates, R-EED-P2 yields still good results with partial known data when the compression rate is doubled.

For small compression ratios, JPEG and JPEG 2000 have a similar performance as our approach (see Fig. 2 and Fig. 3), while PDE-based progressive modes yield noticeably better results for medium to high compression ratios. This observation is not only explained by the different approaches to progressive modes. Other factors are the large file header used by JPEG 2000, and the fact that the baseline R-EED algorithm yields better compression results than JPEG and JPEG 2000.

Figure 3: Images reconstructed from partial files containing the image *trui* compressed with a compression ratio of $76:1$ using JPEG, JPEG 2000, and R-EED, respectively.

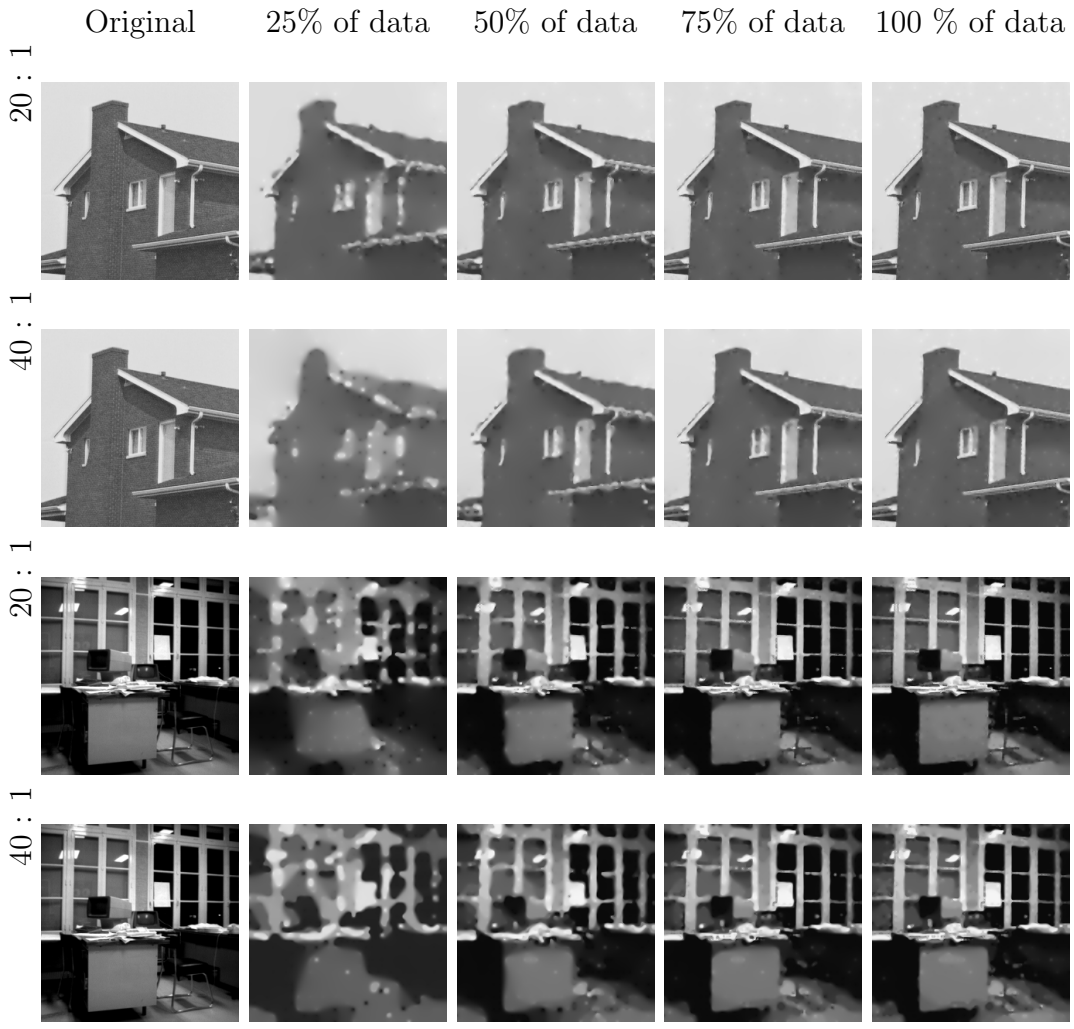


Figure 4: Progressive mode experiments for the test images *peppers*, *house* and *office*. The images were reconstructed with increasing percentages of known data (left to right) and different compression ratios (top to bottom).

4 Region of Interest Coding

4.1 Basic Idea

During the rectangular subdivision phase the R-EED encoder decides in which image region it adds more known data based on local measurements of the mean square error (MSE). The region of interest (ROI) extension to R-EED allows to specify an a priori weighting of different locations in the image. To this end, we introduce local weights $r_{i,j} \in [0,1]$, $i \in \{0,\dots,M\}$, $j \in \{0,\dots N\}$ for a discrete images of size $M \times N$. The ROI-weighted MSE for a reference image $\boldsymbol{f}$ and the corresponding reconstruction $\boldsymbol{u}$ is computed by

$$\mathrm{MSE}_{\mathrm{ROI}}(\boldsymbol{u},\boldsymbol{v}) := \sum_{i,j\in\Omega} \frac{r_{i,j}(u_{i,j}-f_{i,j})^2}{MN}. \tag{5}$$

Subdivision steps with the modified MSE lead to a higher density of known data in regions of interest with high weight r. Consequently, we can reconstruct these regions more accurately than those regions with a low weight. Thereby, the ratio of weights between two regions of interests decides the turning point at which R-EED prefers a smaller quality gain in a high-priority region over a more substantial gain in quality in a low-priority region.
Note that ROI encoding only influences the choice of the inpainting mask by modifying the decisions in the subdivision tree. No additional information needs to be stored and the standard decoder is also applicable for ROI-encoded files. Furthermore, the error-weighting on a per-pixel basis allows to specify multiple regions of interest of arbitrary shape and relative importance.

4.2 Experiments

Examples for applications of ROI coding are given in Figure 5. In medical imaging, ROI coding can be used to store image regions that are important for diagnostic reasons with a local error that is close to zero. The algorithm still reconstructs image regions with a lower priority, albeit with decreased accuracy. Their primary use is to coarsely represent the neighbourhood of high priority ROIs to help the observer with identifying the location of the subregions in respect to the whole imaged object.
Furthermore, ROI coding can provide semantic context to the compression algorithm. As mentioned in Section 2, the subdivision algorithm tries to generate images with a consistent quality in all image regions. Especially for high compression rates, this can lead to suboptimal perceptive image quality since important image structures like faces are treated equally with relatively

unimportant background features. This behaviour is particularity detrimental in cases where unimportant features are costly to compress and thus the quality of the whole image is diminished (see Figure 5). Providing suitable ROI weights either manually or automatically, e.g. with a face recognition algorithm, can substantially improve the perceived quality.

5 Real-Time Video Decoding

In previous publications [19, 18, 17, 15], PDE-based compression codecs focused on achieving the maximum possible quality at a given compression rate. In the following, we examine the performance of R-EED with different numerical solvers and present a framework for real-time video playback with anisotropic diffusion. First, we discuss different numerical approaches for solving the inpainting problem.

5.1 Numerical Solvers

In R-EED, the inpainting problem is described by the elliptic PDE

$$(1 - c_K)\,\mathrm{div}(\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)\boldsymbol{\nabla} u) - c_K\,(u - f) = 0, \tag{6}$$

a special case of the general inpainting equation (1). There are different approaches to discretise and solve this continuous inpainting problem. In the following we discuss solvers from previous implementations of R-EED as well as recent cyclic schemes and compare their suitability for real-time decompression.
One possibility to acquire the inpainted image is to compute the steady state of the parabolic evolution

$$\partial_t u = \mathrm{div}(\boldsymbol{D}\boldsymbol{\nabla} u) \quad \text{on} \quad \Omega \setminus K \times (0, \infty), \tag{7}$$
$$u(\boldsymbol{x}, t) = f(\boldsymbol{x}) \quad \text{on} \quad \partial K \times (0, \infty), \tag{8}$$
$$\langle \boldsymbol{D}\boldsymbol{\nabla} u, \boldsymbol{n} \rangle = 0 \quad \text{on} \quad \partial\Omega \times (0, \infty). \tag{9}$$

This formulation as an initial boundary value problem applies reflecting boundary conditions at the boundary $\partial\Omega$ of the rectangular image domain Ω. Furthermore, the known image data constitutes Dirichlet boundary conditions on K. Experiments show that, independently of the initial value $u(\boldsymbol{x}, 0)$ on $\Omega \setminus K$, the steady state is the same.
A straightforward and easy to implement approach is the explicit discretisation of the parabolic formulation (7). An explicit scheme discretises both

Figure 5: Reconstruction examples with and without region of interest coding for the test images *brain*, *trui* and *nosferatu*. The test images are ordered from top to bottom by increasing compression rate. Bright areas in the ROI-mask have a larger weight in error computation than dark areas. For *brain* the ROI reconstruction is almost perfect in the cerebellum at the cost of reduced quality in other parts. Details such as text are reconstructed more accurately with ROI coding for *nosferatu*, while background structures are omitted. Finally, the ROI in *trui* allows a much higher perceived quality at high compression rates due to better quality in facial features (in particular the eyes).

the temporal and spatial derivatives with suitable finite difference approximations (for more details see [33]). This yields an iterative scheme of the form

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}(\boldsymbol{u}^k)\boldsymbol{u}^k \tag{10}$$

$$\Leftrightarrow \quad \boldsymbol{u}^{k+1} = (\boldsymbol{I} + \tau \boldsymbol{A}(\boldsymbol{u}^k))\boldsymbol{u}^k. \tag{11}$$

Here, $\boldsymbol{A}(\boldsymbol{u}^k)$ is the discretisation of the spatial derivative operators from Equation (7), and τ is the discrete time step size. For stability reasons, the time step size is severely limited and thus the scheme requires a large number of iterations to reach the steady state. However, fast explicit diffusion (FED) [34], a recent cyclic scheme, makes efficient implementations of explicit approaches possible. In particular, FED is well-suited for parallelisation and performs exceptionally well on graphic processing units (GPUs).
Alternatively, one can also solve the elliptic formulation (6) of the inpainting problem directly. Using the same spatial discretisation $\boldsymbol{A}(\boldsymbol{u})$ of the divergence term as before, one obtains

$$(\boldsymbol{I} - \boldsymbol{C}_K)\boldsymbol{A}(\boldsymbol{u})\boldsymbol{u} - \boldsymbol{C}_K(\boldsymbol{u} - \boldsymbol{f}) = 0. \tag{12}$$

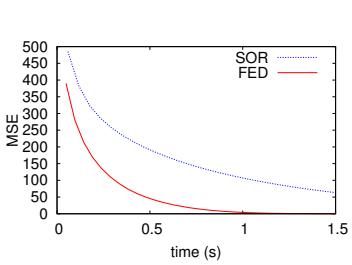
Here, $\boldsymbol{C}_K$ is a quadratic, diagonal matrix that contains the entries of the discrete inpainting mask vector $\boldsymbol{c}_k$. Following the approach of Mainberger et al. [8], Equation (12) can be rearranged into the nonlinear system of equations

$$\underbrace{((\boldsymbol{I} - \boldsymbol{C}_K)\boldsymbol{A}(\boldsymbol{u}) - \boldsymbol{C}_K)}_{=:\boldsymbol{M}(\boldsymbol{u})}\boldsymbol{u} = \boldsymbol{C}_K\boldsymbol{f}. \tag{13}$$

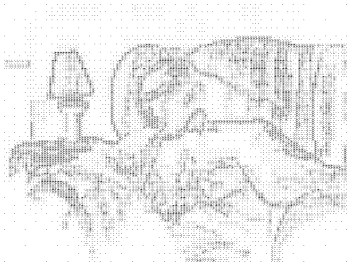
We obtain the solution of this system by the fixed point iteration

$$\boldsymbol{M}(\boldsymbol{u}^k)\boldsymbol{u}^{k+1} = \boldsymbol{C}_K\boldsymbol{f}. \tag{14}$$

For each iteration k, a linear system of equations has to be solved. Previous publications on PDE-based compression [19, 18] use successive over-relaxation (SOR) (see e.g. [35]) for this task. Unlike FED, SOR is in essence a sequential algorithm. Therefore, there is virtually no potential for increasing its performance by parallelisation.
In order to analyse the suitability of solvers for real-time video decoding with R-EED, we compare FED and SOR with respect to their convergence behaviour. To this end, we compute a reference solution for R-EED compressed frames of the movie *Nosferatu* [36] by iterating the explicit scheme until convergence. Each frame has a resolution of 640×480 and we initialise the missing image parts with the average grey value of the known data. For



(a) Convergence Graph



(b) Known Data

(c) FED Inpainting after 0.04s

Figure 6: **Comparison of numerical solvers. (a)** Mean square error in respect to reference solution over time on frame 130254 of the movie *Nosferatu*. **(b)** Known data selected by the R-EED subdivision scheme. Pixels from the set of mask locations K are marked in black, unknown pixels in white. **(c)** EED-Inpainting results after ≈ 0.04 seconds for FED with flat initialisation. Homogeneous regions with a low density of known data are not fully inpainted.

both solvers, Fig. 6 (a) provides the evolution over time of the mean square error (MSE) in relation to the reference inpainting.
On a test system with an *Intel Xeon CPU W3565@3.20GHz* and an *Nvidia Geforce GTX 460*, a parallelised explicit scheme on the GPU typically takes almost 5 minutes to converge. SOR requires around 6 seconds and parallelised FED still needs approximately 2 seconds for convergence. For ill-posed frames, the total inpainting runtime can be even longer.
The comparison in Fig. 6 clearly demonstrates that even the advanced cyclic algorithms with parallelisation cannot achieve real-time video decoding on consumer hardware. Fig. 6 (b) and (c) show the known data and a reconstruction with FED after a runtime of 0.04 seconds. This is the time budget that is available to achieve 25 frames per second. Clearly visible artefacts make these results unsuitable for video playback.

5.2 Smart Initialisation

One way to reduce the runtime of iterative diffusion schemes is to provide a good initalisation. So-called coarse-to-fine approaches [37] already successfully apply this idea. First, they generate pyramid of subsampled versions of the inpainting mask. On the coarsest level, i.e. the smallest representation of the image, inpainting is performed. The upsampled result of this inpainting provides an initialisation for the next finer level of the pyramid. A successive ascend through the finer level of the pyramid finally yields a good initialisation for an inpainting of the full image. Such approaches can be interpreted

as special cases of multigrid solvers [38, 39].
While coarse-to-fine approaches yield a substantial increase in performance, subsampling and diffusion on coarser scales creates some overhead. Furthermore, the benefits of parallelisation are less pronounced on coarse levels due to the smaller number of possible simultaneous calculations. In video sequences, however, there is an additional way to obtain suitable initialisations: In many cases, subsequent frames are very similar. Therefore, the inpainting result of an already processed frame can be reused as an initialisation for the next frame. For groups of pictures (GOPs) that are sufficiently similar, the reuse of inpainting results provides a significant speed-up without any additional computational cost. Consequentially, the core idea of the smart initialisation scheme is to segment a video into such GOPs that allow to reuse as much already computed information as possible.
In some cases, the initialisation with the previous frame is not useful or more difficult, though. In particular at scene transitions or in the case of rapid camera movements, the content of subsequent frames can change drastically and make this initialisation infeasible. Motion can also be a problem if it affects only small areas of the video. It is possible that the initialisation is very good for large parts of the frame, but unsuitable in the vicinity of moving objects. For example, a character rapidly moves a dark object in front of a bright background in the frames 9420 and 9421 of the movie *Nosferatu* (see Fig. 7). This generates almost a worst case initialisation.
However, frames that are partially unsuitable for reuse can still be useful. Initialisations that deviate significantly from the final steady state are only problematic if they occur in areas with a low density of known points. If many known data is given in the corresponding region, their information does not need to be propagated over large distances. Thus, a small number of iterations is sufficient to reach a good approximation to the steady state. Since diffusion only needs a very low amount of known data to reproduce homogeneous image regions, these are prone to initialisation problems. However, in homogeneous regions, a good approximation to the steady state can be achieved by simple and fast linear interpolation.
Therefore, we use a confidence function to decide if parts of initialisation with the last frame should be replaced by linear interpolation results. Let a rectangular image section from the R-EED subdivision be given by the set P of its corner points:

$$P := \{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\}. \tag{15}$$

The confidence into the linear interpolation quality in this rectangle is influenced by two factors: the size of the subimage and its homogeneity. We define

the size factor s_P as the maximum side length of the rectangular region:

$$s_P := \max\{|x_2 - x_1|, |y_2 - y_1|\}. \tag{16}$$

The larger the region is, the larger the expected error of the linear interpolation with the known points from this rectangle. However, this error also depends on the homogeneity of the region. If the rectangular region is perfectly flat, the linear interpolation will be flawless, no matter the size of the region. The more discontinuities appear in the rectangle, the less accurate it becomes.
Therefore, we also incorporate a homogeneity factor h_p. Since only the known image points can be used to determine this aspect of the region, we define h_P as

$$h_P := \max_{(x,y)\in P} f(x,y) - \min_{(x,y)\in P} f(x,y) \tag{17}$$

For large maximal deviations of the known grey values in the region, less accurate results can be expected from the linear interpolation.
A multiplicative combination of the size and homogeneity factors yields the overall confidence function c_P:

$$c_P := \psi(s_P) \cdot h_P. \tag{18}$$

Here ψ is a truncated quadratic function with a threshold parameter ξ (default value $\xi = 30$) that also rescales the range of the size factor to the interval $[0, 1]$:

$$\psi(x) = \min(x^2, \xi)/\xi.$$

The threshold ξ and the rescaling avoids that the size factor dominates the confidence measure. This can happen since the size factor is limited to the range $[0, 255]$ while the image size is not limited. For image sections with a height or width smaller than ξ, the homogeneity factor is attenuated. For larger sections, it has the full impact.
The two examples in Figure 7 illustrate that the use of the confidence-dependent linear initialisation in smart initialisation offers significant improvements over a pure reuse initialisation. Experimental results on a large number of images are discussed in Section 5.5.

5.3 An EED-based Framework for Video Playback

In the following, we describe a framework for real-time video reconstruction with anisotropic diffusion inpainting. In this paper we solely focus on the

aspect of real-time reconstruction with smart initialisation. To this end, we first present a wrapper that extends R-EED to video sequences and provides the necessary structures for frame reuse. We do not address compression quality and apply frame-by-frame compression with standard R-EED. In future publications, the R-EED wrapper can act as a starting point for true PDE-based video encoders that explicitly exploit temporal redundancies in the video. The second part of the video player framework is the real-time decoder that we discuss in detail in Section 5.4.

The core structure of our framework relies on a segmentation of the video in groups of pictures (GOPs). The primary purpose of these GOPs is to define regions where smart initialisation is feasible. Therefore, we employ scene change detection to create a temporal segmentation.

Smart initialisation always requires the reconstruction of the previous frame to decode the current frame. Therefore, the frames in a GOP require successive decoding starting with the first frame of the GOP. This implies that random access in the video (e.g. for fast forwarding or chapter selection) is limited to the beginning of GOPs, similar to common video compression codecs such as the ones from the MPEG and H26X family. Due to this constraint, we also limit the maximal number of frames N in a GOP. For practical purposes, we choose $N = 32$ in this publication.

Each GOP is essentially a collection of up to N R-EED encoded images. Since by definition the frames in a GOP have similar content, we can choose a common R-EED contrast parameter λ and a common quantisation parameter q for the whole GOP. All other R-EED parameters are identical for the whole video and thus only the inpainting mask has to be stored for individual frames. For sufficiently similar frames, it is possible to reuse the mask positions or even the whole inpainting mask including the grey values without a significant loss in compression quality. Therefore, we introduce four different frame types which we categorise by the similarity to previous frames. Depending on the frame type, the framework stores different parts of the inpainting mask.

- **GOP start (type 1):** Always contains the binary tree that encodes the known point positions and the grey values of the inpainting mask. This guarantees that all information to reconstruct this frame independently from other frames is available. The framework assigns type 1 to a frame if the maximum number N was reached in the previous GOP or the difference to the previous frame exceeds a threshold $t_1 > 0$ (default value: $t_1 = 15$). Additionally, we create a new GOP if the difference between the reference steady state and the reconstruction exceeds a threshold $t_{\text{ref}} > 0$ (default value: $t_{\text{ref}} = 5$). This additional constraint

avoids error spikes at ill-posed images, e.g. those that contain interlacing artefacts.

- **No mask reuse (type 2):** For this type, the framework does not reuse point positions, just the initialisation and R-EED parameters. Just like for a type 1 frame, we store both the binary tree and the quantised grey values. The framework assigns type 2 if the difference to the previous frame exceeds the threshold $t_2 \leq t_1$ (default value: $t_2 = 5$).

- **Tree reuse (type 3):** The point positions from the previous frame remain unchanged, but we store different quantised grey values for these positions. The framework assigns type 3 if the difference to the previous frame exceeds the threshold $t_3 \leq t_2$ (default value: $t_3 = 1$).

- **Full mask reuse (type 4):** We consider this frame to be fully redundant and no additional data is stored. The framework assigns type 4 if the difference to the previous frame does not exceed the threshold t_1.

Just as in the regular R-EED scheme, we store the mask positions as a binary sequence of splitting decisions and a minimum and maximum tree depth. An entropy coder saves the grey values efficiently. Furthermore, a GOP header contains all information that is needed to decode the GOP frames. It provides the content-dependent R-EED parameters λ and q, as well as a list of frame types. Additionally, the header includes a list with the lengths of each GOP data set to enable separation of the frame data for all possible entropy coders. Finally, all of the GOP data must be included in a single video file. This container format consists of a global video header and a concatenation of all GOP data. The global header contains all parameters that do not have to be adapted to the specific image content: video resolution, total number of frames, a colour flag, the entropy encoding type, the point pattern for rectangular subdivision, and the progressive mode parameters. In order to allow random access to the GOP start frames, the header also includes the length of all GOPs. This enables the video player to jump to arbitrary GOP data.
For the global header and all GOP headers, the parameters are encoded directly as binary numbers of custom length. The frame and GOP lengths are stored efficiently with Golomb coding [40].

5.4 Architecture of the Real-Time Decoder

Since the main goal of our framework is real-time video playback, the decoder has to provide reconstructions of a rate of 25 frames per second. This implies

that each reconstruction must be available in a time window of 0.04s. Consumer class GPUs can perform around 100 iterations with 5 linear updates for FED solvers in this time.
Additionally, there is some overhead due to file operations, entropy decoding, smart initialisation, and data transfer between CPU and GPU. Moreover, at GOP start frames, there is no initialisation with the reconstruction from the last frame available. In order to minimise overhead and handle start frames, we propose a multi-threaded decoder with caching that makes use of widely available multicore CPUs.
The decoder relies on four distinct threads that operate in parallel and only communicate over a system of caches for initialisation and reconstructed image data.

- **Loader:** The loader parses header data and extracts compressed data for a single frame from the video file. It rebuilds the inpainting mask from stored tree data and decompresses the entropy coded grey-value data. Finally, it computes the linear interpolation for smart initialisation. In order to use times of low CPU usage efficiently, a *GOP init cache* collects the complete inpainting mask and computed linear initialisation for multiple GOP start frames ahead of time. Similarly, the *loader cache* provides the initialisation data for all other frames. These caches are separated due to their different amount of localisation. The initialisations for GOP start frames correspond to video parts that have a larger temporal distance to the currently displayed frames in order to enable random access and avoid quality loss through inpainting activity spikes in video regions with rapid scene changes. In contrast, precomputed initialisations for other frames only cover GOPs in a close vicinity to the current frame.

- **GOP-start Handler:** Another thread is entirely dedicated to generating an initialisation for GOP-start frames with coarse-to-fine inpainting on the CPU. It reads the mask and known grey values from the GOP init cache, reconstructs the start frames and stores them in a third cache, the *GOP start cache*. In order to avoid playback pauses, slowdowns or dropped frames, the start handler dynamically adapts the number of iterations and nonlinear updates of the numerical solver to the number of cached frames. If the cache is in danger of running empty, the number of iterations is reduced, if it is sufficiently full, the number of iteration is increased. Thereby, fluent playback is maintained at the cost of a slight degradation in reconstruction quality for the GOP start frames. This is e.g. relevant for user interaction with random access or

temporary load spikes of the CPU due to other applications or ill-posed sections in the video.

- **Reconstructor:** The reconstructor is responsible for the final inpainting of the frames. Its main task is to transfer initialisation data to the GPU, perform FED inpainting and store the finished reconstruction in the *reconstruction cache*. For all frames except GOP start frames, it also generates the final initialisation. According to the confidence measure for the smart initialisation, it combines the linear interpolation data from the loader cache with the last reconstruction from the reconstruction cache. Just as the GOP-start handler, the reconstructor also adapts the number of iterations and nonlinear updates to the number of cached frames to guarantee fluent playback (if the cache is nearly empty) or increase the fidelity (if the cache is sufficiently full).

- **Presenter:** The presenter handles all user interaction and video-playback with the OpenGL toolkit *GLUT*. It reads the decompressed images from the reconstruction cache at a rate of 25 frames per second.

The architecture of the decoder allows to dedicate more time to the actual inpainting by focussing the time-critical GPU interpolation to a single thread and distributing all other tasks to the three other threads. Additionally, the caching system even allows to increase the inpainting runtime above 0.04s if a larger time budget is available. This can happen if time is saved due to the reuse of whole frames (frame type 4) or if the user pauses the video.

5.5 Experiments

In the following we present experiments that assess the quality of the decoded images with different initialisation strategies. Since the decoded results should approximate the steady-state as accurately as possible, we compare them against reference results.
These reference inpaintings result from an explicit scheme with FED acceleration. As a stopping criterion, we used the maximal per-pixel change after a stopping time of $T = 10000$ with 20 nonlinear updates. For the initialisation $\boldsymbol{u}^k$ and the corresponding reconstruction $\boldsymbol{u}^{k+1}$ after a diffusion time of 10000 we define the per-pixel change

$$d(\boldsymbol{u}^k, \boldsymbol{u}^{k+1}) := \max_{i,j} \{|u_{i,j}^k - u_{i,j}^{k+1}|\}. \tag{19}$$

For $\boldsymbol{u}^0$ we use a flat initialisation with the average grey value of the known data. The FED inpaintings with stopping time $T = 10000$ are then iterated until $d(\boldsymbol{u}^k, \boldsymbol{u}^{k+1}) < 0.001$.

Table 1: Deviation of inpainting results from the corresponding reference reconstructions. Different initialisation strategies with a Fast Explicit Diffusion solver are compared in respect to the reference MSE on the movie Nosferatu. The default parameters from Section 5.3 were used for encoding. The maximum error is the same for reuse and smart initialisation due to the application of the t_{ref} threshold (see Section 5.3). Note that the main benefit of smart initialisation is the removal of local artefacts. Nevertheless, there is also an improvement in the average error.

Strategy	Flat (FED)	Reuse (FED)	Smart (FED)
Maximum Error	16788.51	4.99	4.99
Average Error	461.46	0.20	0.19
Error < 0.5	1.39%	91.42%	91.60%
Error < 1	2.51%	95.31%	95.68%
Error < 5	4.91%	100%	100%
Error < 10	6.94%	100%	100%

All tests were performed on an *Intel Xeon CPU W3565@3.20GHz* with an *Nvidia Geforce GTX 460*. A restored version of the classical movie Nosferatu [36] from 1922 acts as a test file. The frames in Fig. 7 and Fig. 6 are copyrighted by the Friedrich-Wilhelm-Murnau-Foundation and are published with their consent. It has 141680 grey scale frames at a resolution of 640×480. For our testing purposes, the full movie was encoded at a compression rate of 20:1 with the default encoder parameters from Section 5.3. For the decoder, we used an FED scheme with 5 nonlinear updates and 20 iterations per cycle for all tested initialisation strategies.

In Table 1 the deviation of the inpainting results from the reference solutions are displayed for several initialisation strategies. We measure this deviation in terms of the mean square error (MSE) between the reference reconstruction and the decoder reconstructions with different initialisation strategies.

Using a flat initialisation is completely infeasible for video playback. An average error of 37.82 suggests severe artefacts in many frames. In contrast, the reuse heuristic already decreases the average MSE to 0.25. In fact, 95.31% of the frames have a reference MSE below 0.5 and are visually indistinguishable from the reference solution in practice. We also compare against our smart initialisation strategy, which is mainly used to avoid the creation of local artefacts in worst case scenarios as described in Section 5.2. The results in Table 1 also demonstrate that smart initialisation does not introduce

significant negative side effects. In contrary, the average error is decreased by 5%.
For the reuse and smart initialisation strategies, there are still some frames with an MSE close to the GOP reference threshold $t_{\text{ref}} = 5$. However, their number is relatively small and the visual difference is hard to spot at 25 frames per second.
In total, the results show that real time playback of R-EED-encoded videos is possible on consumer hardware. In particular, no compromises have to be made in regard to encoding quality in order to achieve a good approximation of the reference solution. Due to smart initialisation, no additional data has to be stored in problematic regions.

6 Conclusion

With the three extensions introduced into this paper, we have made a strong case for the suitability of R-EED for real-world applications. In particular, the real-time decoding capabilities of our video player framework demonstrate that even though PDE-based methods are in general more computationally-intensive than transformation-based methods, a clever use of available resources allows surprising performance on consumer hardware. These extensions mark the first step of an evolution of PDE-based compression schemes from the proof-of-concept stage to fully grown codecs with relevance for practical applications.
In our future work, we plan to investigate efficient video *encoding* algorithms for the PDE-based video *decoding* algorithm presented here. To further enhance the performance of the proposed progressive modes, especially for very small file parts, we plan to look into progressive modes that only store a part of the tree structure in the file header, while the remainder is stored closer to the end of the file. Another line of research is to transfer our progressive mode and region of interest coding scheme to other PDE-based compression approaches for 2-D, 3-D and video data.

References

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

[2] D. S. Taubman, M. W. Marcellin (Eds.), JPEG 2000: Image Compression Fundamentals, Standards and Practice, Kluwer, Boston, 2002.

[3] S. Carlsson, Sketch based coding of grey level images, Signal Processing 15 (1988) 57–83.

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

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

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

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

[8] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann, B. Doerr, Optimising spatial and tonal data for homogeneous diffusion inpainting., in: A. Bruckstein, B. ter Haar Romeny, A. Bronstein, M. Bronstein (Eds.), Proc. Third International Conference on Scale Space and Variational Methods in Computer Vision, Vol. 6667 of Lecture Notes in Computer Science, Springer, Berlin, 2011, pp. 26–37.

[9] J. Gautier, O. L. Meur, C. Guillemot, Efficient depth map compression based on lossless edge coding and diffusion, in: Picture Coding Symposium, Kraków, Poland, 2012, pp. 81–84.

[10] Y. Li, M. Sjostrom, U. Jennehag, R. Olsson, A scalable coding approach for high quality depth image compression., in: 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, Zurich, Switzerland, 2012, pp. 1–4.

[11] S. Hoffmann, M. Mainberger, J. Weickert, M. Puhl, Compression of depth maps with segment-based homogeneous diffusion, in: A. Kuijper, K. Bredies, T. Pock, H. Bischof (Eds.), Scale-Space and Variational Methods in Computer Vision, Vol. 7893 of Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 319–330.

[12] T. F. Chan, H. M. Zhou, Total variation improved wavelet thresholding in image compression, in: Proc. Seventh International Conference on Image Processing, Vol. II, Vancouver, Canada, 2000, pp. 391–394.

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

[14] D. Liu, X. Sun, F. Wu, Edge-based inpainting and texture synthesis for image compression, in: Proc. 2007 International Conference on Multimedia and Expo, Beijing, China, 2007, pp. 1443–1446.

[15] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, H.-P. Seidel, Towards PDE-based image compression, in: N. Paragios, O. Faugeras, T. Chan, C. Schnörr (Eds.), Variational, Geometric and Level-Set Methods in Computer Vision, Vol. 3752 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 37–48.

[16] J. Weickert, Theoretical foundations of anisotropic diffusion in image processing, Computing Supplement 11 (1996) 221–236.

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

[18] C. Schmaltz, J. Weickert, A. Bruhn, Beating the quality of JPEG 2000 with anisotropic diffusion, in: J. Denzler, G. Notni, H. Süße (Eds.), Pattern Recognition, Vol. 5748 of Lecture Notes in Computer Science, Springer, Berlin, 2009, pp. 452–461.

[19] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert, A. Bruhn, Understanding, optimising, and extending data compression with anisotropic diffusion, International Journal of Computer Vision 108 (3) (2014) 222–240.

[20] CompuServe Incorporated, Graphics interchange format (tm). a standard defining a mechanism for the storage and transmission of raster-based graphics information, W3C specification, W3C, `http://www.w3.org/Graphics/GIF/spec-gif87.txt` (Jun. 1987).

[21] T. Boutell, RFC 2083: PNG (portable network graphics) specification version 1.0, status: INFORMATIONAL. (Jan. 1997).
URL `ftp://ftp.internic.net/rfc/rfc2083.txt`

[22] N. Kaur, A review of region-of-interest coding techniques of JPEG2000, IJCA Special Issue on Computational Science - New Dimensions & Perspectives 1 (2011) 35–40.

[23] J. S. Pawadshetty, J. W. Bakal, JPEG 2000 region of interest coding methods, International Journal of Engineering Research and Applications 3 (1) (2013) 1184–1188.

[24] Q. Gao, Low bit rate video compression using inpainting PDEs and optic flow, Master's thesis, Dept. of Computer Science, Saarland University, Saarbrücken, Germany (2008).

[25] H. Köstler, M. Stürmer, C. Freundl, U. Rüde, PDE based video compression in real time, Tech. Rep. 07-11, Lehrstuhl für Informatik 10, Univ. Erlangen–Nürnberg, Germany (2007).

[26] K. Baum, GPGPU supportet diffusion-based naive video compression, Master's thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany (2013).

[27] C. Schmaltz, N. Mach, M. Mainberger, J. Weickert, Progressive modes in PDE-based image compression, in: Picture Coding Symposium, IEEE, Piscataway, NJ, 2013, pp. 233–236.

[28] J. Weickert, M. Welk, Tensor field interpolation with PDEs, in: J. Weickert, H. Hagen (Eds.), Visualization and Processing of Tensor Fields, Springer, Berlin, 2006, pp. 315–325.

[29] M. Mahoney, Adaptive weighing of context models for lossless data compression, Tech. Rep. CS-2005-16, Florida Institute of Technology, Melbourne, Florida (Dec. 2005).

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

[31] D. A. Huffman, A method for the construction of minimum redundancy codes, Proceedings of the IRE 40 (1952) 1098–1101.

[32] E. Bae, J. Weickert, Partial differential equations for interpolation and compression of surfaces, in: M. Daehlen, M. Floater, T. Lyche, J.-L. Merrien, K. Mørken, L. L. Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Vol. 5862 of Lecture Notes in Computer Science, Springer, Berlin, 2010, pp. 1–14.

[33] J. Weickert, M. Welk, M. Wickert, L2-stable nonstandard finite differences for anisotropic diffusion, in: A. Kuijper, K. Bredies, T. Pock, H. Bischof (Eds.), Scale Space and Variational Methods in Computer Vision, Vol. 7893 of Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 380–391.

[34] S. Grewenig, J. Weickert, A. Bruhn, From box filtering to fast explicit diffusion, in: M. Goesele, S. Roth, A. Kuijper, B. Schiele, K. Schindler (Eds.), Pattern Recognition, Vol. 6376 of Lecture Notes in Computer Science, Springer, Berlin, 2010, pp. 533–542.

[35] R. A. Varga, Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, 1962.

[36] F. W. Murnau (director), A. Grau, E. Dieckmann (producers), Nosferatu, eine Symphonie des Grauens, restored Version courtesy of the Friedrich-Wilhelm-Murnau foundation, Wiesbaden (1922).

[37] F. Bornemann, P. Deuflhard, The cascadic multigrid method for elliptic problems, Numerische Mathematik 75 (2) (1996) 135–152.

[38] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Mathematics of Computation 31 (138) (1977) 333–390.

[39] W. Hackbusch, Multigrid Methods and Applications, Springer, New York, 1985.

[40] S. Golomb, Run-length encodings, IEEE Transactions on Information Theory 12 (3) (1966) 399–401.

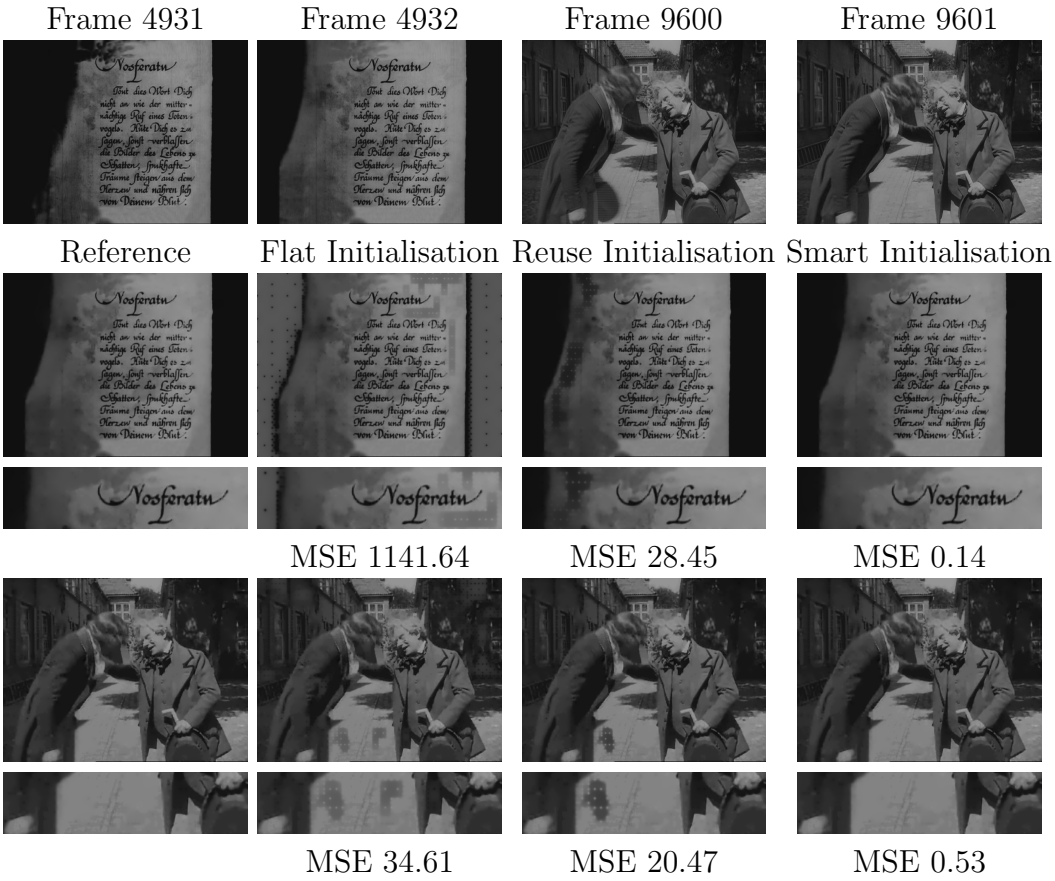


Figure 7: Reconstruction of scenes with quick movement based on different initialisations (compression ratio 20:1). The first row shows the original frames 4932 and 9601 and their predecessors. From Frame 4931 to 4932 a book page is flipped. Frames 9600 and 9601 contains quick movement of a hat in the hand of the character to the left. The first columns displays the reference inpainting, while the second column uses a flat, black initialisation. The results in the third column were obtained using the last frame as an initialisation and the last column is based on a smart initialisation that combines information from the last frame and linear interpolation results based on a confidence measure. Smart initialisation provides the lowest MSE for the total reconstruction.