

Variational decomposition of image data from DjVu encoded files

Martin Holler, Kamil S. Kazimierski

Abstract—The DjVu file format and image compression techniques are widely used in the archival of digital documents. Its key ingredients are the separation of the document into foreground and background layers and a binary switching mask, followed by a lossy, transform-based compression of the former and a dictionary-based compression of the latter. The lossy compression of the layers is based on a wavelet decomposition and bit truncation, which leads, in particular at higher compression rates, to severe compression artifacts in the standard decompression of the layers.

The aim of this paper is to break ground for the variational decompression of DjVu files. To this aim, we provide an in-depth analysis and discussion of the compression standard with a particular focus on modeling data constraints for decompression. This allows to carry out DjVu decompression as regularized inversion of the compression procedure. As particular example, we evaluate the performance of such a framework using Total Variation and Total Generalized Variation regularization. Furthermore, we provide routines for obtaining the necessary data constraints from a compressed DjVu file and for the forward and adjoint transformation operator involved in DjVu compression.

I. INTRODUCTION

VARIATIONAL methods constitute a universal and highly appreciated tool in image processing [1]. In many cases, they are the key to obtaining state-of-the-art results for diverse fields of application such as medical imaging, microscopy and also image decompression. The aim of this work is to break ground for the variational decompression of DjVu files. DjVu is a heavily used file format designed primarily to store scanned documents, promising smaller files than PDF [2].

Variational DjVu decompression comprises the minimization of an appropriate regularization functional subject to suitable data constraints that depend on the compressed file. Obtaining the latter requires a thorough analysis of the DjVu compression procedure which will be carried out in this work. For the former, any regularization functional can be employed in principle and we provide exemplary results with classical and state-of-the-art approaches. The practical realization of variational DjVu decompression, further, requires reading the data constraints from a compressed file and the availability of the forward and (typically) the adjoint transform operator mapping from image to data space, for which appropriate MATLAB implementations are provided in the supplementary material of this paper.

The DjVu standard generally is designed for handling the document to be compressed in three different layers, a fore-

and background layer and a binary switching mask. While the compression of the mask is achieved mainly by exploiting repeating patterns, such as letters, with a dictionary-based encoding, compression of the layers relies on a deliberate loss of data that is less relevant image quality. This is done essentially via a wavelet transform and subsequent bit truncation, which leads, at higher compression levels, to strongly visible artifacts in the images. Our work is focused on inverting the compression of these layers.

Compression of the fore- and background layers in DjVu in general consists of two major steps: The first one comprises a color space transform, a subsequent wavelet decomposition of the resulting color channels and in a final part a decimation step, the latter being responsible for the ‘lossy’ property of the image compression. In a second step, an entropy-based encoding is applied to the serialized data of the first step. This lossless part of the encoding is implemented via (context adaptive, binary) arithmetic coding. With the decoding in mind, the only problematic part of the encoding chain is the decimation, since, obviously, there are many data which, when decimated, return the same result, i.e., the decimation is not an invertible process.

The standard decoding process consists of an entropy decoding followed by an inverse wavelet transform and an inverse color transform. Virtually, the decimation step of the encoding is ignored, and the truncated wavelet coefficients are taken as the ground truth. This choice, however, is somewhat arbitrary, in particular it is carried out regardless of any additional information on the geometric properties of the image. In contrast to that, variational decompression allows to incorporate such information via regularization and hence to improve inversion of the decimation.

Within the encoding/decoding chain, the entropy decoding and the color transform can be regarded as fixed pre- and post-processing steps of our approach. The color transform in DjVu (as is usual in image compression) is a pixel-wise/local and invertible transform. Therefore any local geometric properties of the image can be equivalently formulated in the original as well as in the color-transformed space. Further the entropy decoding can be done as preprocessing as it is assumed to be invertible too.

Variational image decompression scheme. First, the decimated wavelet coefficients (of the original image) are obtained by entropy decoding. Together with a coefficient-dependent decimation-strength parameter (which we obtain from the file) this allows to characterize a set D of admissible undecimated wavelet coefficients, i.e., a set of images in the wavelet domain which - after decimation - have the same decimated wavelet coefficients as the original one. The main computational step

is now to find an image for which the wavelet-transform lies in D and which at the same time admits a given image model optimally. This is done by minimizing an appropriate regularization functional that realizes the image model subject to the constraints resulting from D . Finally, from that optimal image (in the color space) an image (in the original space) is obtained by inverting the color transform.

For a concrete realization of variational DjVu decompression we employ Total Variation (TV) [3] and Total Generalized Variation (TGV) [4] regularization as examples of classical and state-of-the-art regularization approaches, respectively, and also compare to non-local TV regularization.

Related work. There are, to the best knowledge of the authors, no other works available that approach DjVu decompression via variational modeling and optimization. Regarding the underlying problem structure of DjVu image decompression, variational JPEG 2000 decompression is closely related. Even though the coefficient encoding and wavelet types are different, the optimization problem for variational DjVu image decompression itself is in fact a particular case of the general setting analyzed in [5], [6], dealing with applications to JPEG and JPEG 2000 decompression. Furthermore, the works [7], [8], [9], dealing with wavelet inpainting, cover a related setting and we provide a comparison to the approach of [8], which uses non-local TV regularization. More generally, considering DjVu image decompression as variational image dequantization in a basis-transform-domain, also variational JPEG decompression is related and we refer to [10], [11], [12] for comparable approaches. For decompression aspects dealing with modern video coding we refer to [13], [14] and the references therein.

Independent of such conceptually related methods, however, we stress that going from optimization subject to constraints on a wavelet decomposition to the variational decompression of actual DjVu files as elaborated in this work is a non-trivial step that has not been addressed previously in literature. In particular, as mentioned above, this step requires an in-depth analysis of DjVu data handling, appropriate file reading routines and efficient implementations in particular of the adjoint of the non-standard wavelet operator used in DjVu.

Paper outline. For the construction of the set D , a thorough knowledge of the DjVu format, in particular its serialization and decimation steps, is necessary. To the authors best knowledge this information is not available in the literature beyond the original format specification (cf. e.g. [15]). Therefore we provide a thorough overview over the specific properties of the DjVu format in the next section. In Sections III and IV we then exemplarily describe the setup and numerical solution for variational DjVu decompression using TGV regularization, noting that a TV regularized variant can be considered as a simpler case. The paper closes with a presentation of results on variational DjVu decompression using different regularization approaches, comparing also to results obtained with state-of-the-art denoising methods.

II. DJVU

The DjVu image compression technology and file format was developed in the second half of the nineties [2]. A detailed

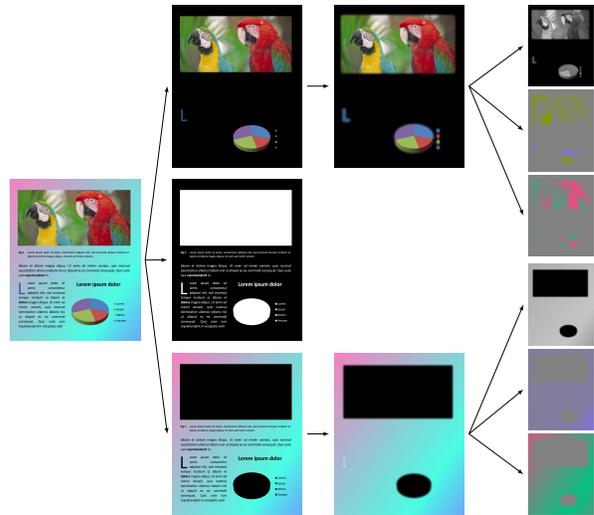


Fig. 1. Decomposition of a single DjVu page into layers and subsequent decomposition of layers into color channels.

specification of the pure image format (version 2) and the variously extended version 3 is available online at [16], [17]. An open reference implementation is available at [15].

A DjVu file is a container based on the EA IFF 85 [18]. It consists of chunks aligned on 16-bit words. Beginning with version 3 the DjVu supports multi-page documents. On a structural level a single page consists of three principal parts, the foreground image, the background image and a mask indicating the foreground, see Figure 1. These parts can also be enriched by additional data like thumbnails, annotations or OCR data. In the simplest case, the foreground, background and mask parts of a page are raster images both having the same dimension as the page itself. The foreground and background can be either gray-scale or color images, whereas the mask is always bi-tonal. The foreground part depicts fine features of the page like figures, tables and text, whereas the background part consist of coarse parts of the page like paper structure or smooth background (e.g. color gradients). The mask is then used to locally switch between the foreground and the background image.

The DjVu specification allows for great flexibility in handling the three principal components. The mask can either be stored in DjVu native, lossless JB2 format or as CCIT Fax Group 4 (G4/MMR) image. The JB2 format decomposes the bi-tonal mask into a list/dictionary of smaller shapes which are then be encoded directly or as refinements of other shapes. Finally the dictionary is serialized and compressed by an DjVu native entropy coder (the so called ZP coder or Z' coder). The foreground and background data can be either stored in the DjVu native IW44 format or as a JPEG image. It is worth noting that the DjVu image model of separating a single page into foreground, background and mask needs an appropriate segmentation, which the standard assumes to be given.

While the concept of page separation into different layers reflects the focus of DjVu to scanned documents, the main

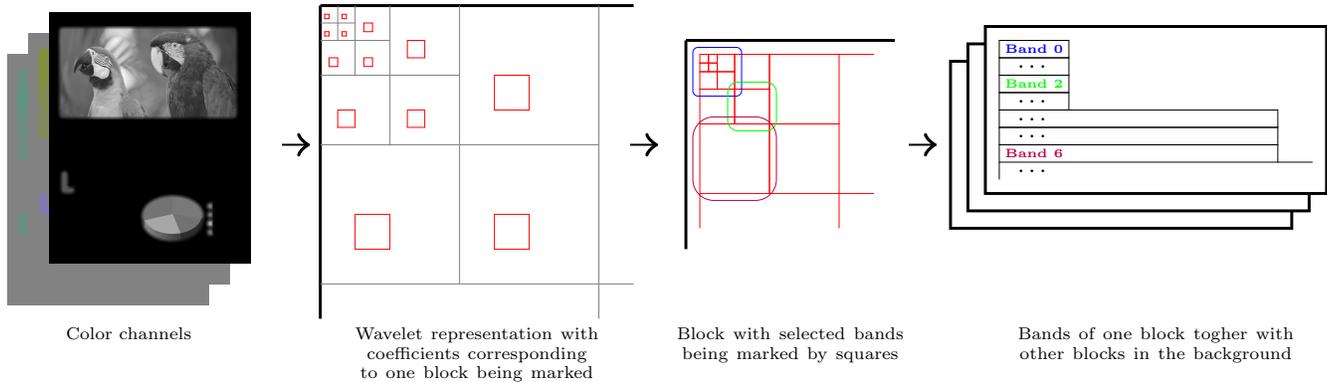


Fig. 2. Serialization layout for IW44 coefficients.

ingredients of its compression performance are an efficient storage of the bi-tonal mask and the two image layers representing fore- and background information. For the latter, a loss of data is explicitly tolerated by using lossy transform-based image encoding, which naturally leads to compression artifacts at higher compression rates. The focus of this work is to avoid such compression artifacts by using variational methods to invert the lossy encoding of the image layers within the DjVu standard. We will focus on the DjVu native IW44 format for images, however, similar techniques can be carried out for JPEG images (see [5], [6]).

In a nutshell, the information on the fore- and background layers is stored via the coefficients of a wavelet representation with respect to the Dubuc-Deslauriers-Lemire (DDL) $(4, 4)$ wavelets [19]. To achieve compression, the wavelet coefficients are re-ordered according to their expected importance for visual image quality, see Figure 2, and subsequently encoded via a bit-level-wise encoding stream, see Figure 3. This produces 200 so called slices, where each slice essentially contains the information of one bit level of a set of coefficients. The number of actually stored slices then defines the compression rate for the image and determines the precision up to which the wavelet coefficients representing the image are known. A detailed description of the layer encoding and decoding is provided in the following.

A. DjVu layer encoding

For compression, the fore- and background layer of each page within a DjVu file is transformed to the YCbCr color space that separates brightness and color information. Subsequently, each color channel is represented in terms of the wavelet coefficients of the 5-level (DDL) $(4, 4)$ wavelet transform and encoded separately.

The main container for storage of wavelet coefficients is a block of 1024 coefficients representing roughly a patch of 32×32 pixels in the original image. The wavelet coefficients of a single block are selected as a set of spatially neighboring coefficients from all wavelet bands. The coefficients of each block are then grouped to 10 different bands, which are related but do not agree with the wavelet bands, see Figure 2.

Every band is then subdivided into a variable number of so called buckets (i.e. containers of 16 coefficients). The first four

bands have 1 bucket each, the following 3 bands consist of 4 buckets each and finally the last three bands have 16 buckets each. The partitioning of bands into buckets is relevant for the update of contexts within decoding, as described later, but will not be needed for the moment.

The above described *physical* partition of the wavelet coefficients of an image in blocks and bands is then used for a *logical* partition by means of bit-slicing. Each single wavelet coefficient is represented as a tuple of a 1-bit sign and 15-bit mantissa. (Notice that this is *not* equivalent to a 16-bit signed int). In order to be able to give less importance to some coefficients when employing a bit-level wise encoding, the 15 bits representing the coefficient magnitude are virtually extended to 20 bits, by introducing empty bit levels. A slice then encodes one bit level of one band of all blocks, e.g., slice 0 encodes the highest bit of all coefficients in the first band of all blocks, see Figure 3. As there exist 10 bands in each block and each coefficient has 20 (virtual) bit-levels, the bit-slicing generates 200 slices in total. Due to the ordering in which the coefficients contribute to the slices, the importance of slices for visual image quality is assumed to decrease for subsequent slices. The number of stored slices is the main compression parameter of the IW44 format.

Within the DjVu file, the streams of slices of all channels are stored in so called cycles (group of one slice for every channel). Doing so, the format also allows for a delay of both chroma channels with respect to the luminance channel. This delay is a second parameter which can be employed for compression purposes.

B. DjVu layer decoding

Given the encoded DjVu file, an overview of the main decoding loop is given as follows: The image is decoded in cycles, which consist of the decoding of a given slice for every channel. Every slice is then decoded block by block. Since a single slice always deals with one band of coefficients only, the decoding of a bit-slice for a band of coefficients within a single block is the main part of the decoding procedure. Within the decoding process, the standard DjVu decoding process estimates the full precision wavelet coefficients by taking the center of their admissible range defined by the

low precision data. The image is then obtained by an inverse wavelet transform on these estimated coefficients.

In the following paragraphs, we will describe the IW44-based DjVu image decompression procedure, which is split into *band decoding*, *wavelet reconstruction* and *color reconstruction*, in more detail.

Band decoding. Following the overview of the decoding loop as described above, this paragraph deals with the decoding of a single bit slice of a given band of a single block of coefficients. To this aim, remember that each band is divided into 1, 3 or 16 buckets, each containing 16 coefficients. Accordingly, the bit slice decoding of a band is subdivided into a bit slice decoding of the coefficients of each bucket.

Now given the encoded bit stream corresponding to a bit slice of the current band, the decoder naturally has to interpret each bit of this stream. For storage and computation efficiency, the meaning of each bit depends on the current state of the partly decoded coefficients, buckets and bands. For example, depending on the state of the current bucket, a single bit of the stream can either encode that the current bit level of all coefficients in the bucket (having a certain state) contains 0, or it can encode the value of the current bit level of a single coefficient. To account for that, the decoder tracks the current state of each coefficient, bucket and band using *potential* and *active* flags.

Regarding these flags, first remember that each wavelet coefficient is represented as a tuple of a 1-bit sign and 15-bit mantissa. As explained in the last section (cf. also Figure 3) the bits of the mantissa are decoded beginning with the most significant bit down to the least significant bit. Within this process the sign bit is decoded together with most significant 1 bit. Consequently, no sign bit is decoded for the wavelet coefficient value of 0. A single coefficient at a current state is now called potential, if all bit-levels decoded so far contained only 0, i.e., the sign-bit has not yet been decoded. Otherwise, if already one 1 bit has been decoded, the coefficient is called active. Similarly, a bucket and band is set potential/active if it contains at least one potential/active coefficient. In contrast to the coefficients, the potential and active flags for buckets and bands can be set at the same time.

As initial step for decoding of a bit slice of the current band, all relevant coefficients are identified as active or potential. Further, all buckets and the band itself are marked as active and/or potential. The decoding is then carried out in four steps.

The first step deals with the question if there are coefficients in the current band which will transition from potential to active in this band decoding. The answer to this question is stored in a flag. This flag is always set for bands 0-6. For bands 7-9, if the band is active the flag is set otherwise the flag is decoded from the bit stream at the current pointer position.

The second step is the analogue of the first step at the bucket level. If the flag from last step is set, then for every potential bucket a flag is decoded from the bit stream, determining if the values of the current bit level of all potential coefficients contains one non-zero entry or not.

The third step then deals with the potential to active transition at the coefficient level. For all potential coefficients in buckets with the flag from the second step set, the current

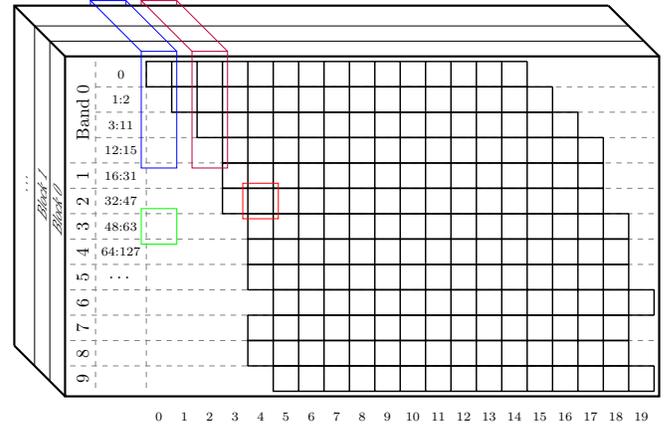


Fig. 3. Band-wise bit-slicing. A bitslice (colored highlights) are selected bits of coefficients within a selected band throughout all blocks of the image.

bit is decoded from the bit stream. If the bit is 1, i.e., the coefficient transitions to the active state, then additionally the sign bit is decoded. The bit for potential coefficients in buckets for which the flag from the last step was not set are set to 0.

Finally, in the last step, the current bit for all active coefficients is decoded from the stream, completing the decoding of a bit-level of the current band. This decoding procedure is then repeated for all bands, blocks and slices and color components until the end of the bit-stream.

Wavelet and color reconstruction. The result of the band decoding is a sequence of blocks with 1+15 bit-tuples. Depending on how many bit-levels have been decoded, these tuples may only be known up to a certain precision, i.e., represent an interval of possible values. Such tuple are then replaced by the middle of the corresponding interval and all tuples are converted to a 16-bit number.

The blocks of coefficients (now represented by 16-bit numbers) are inherently linear data-structures. In the next step they are deserialized to 32 x 32 blocks, which are then copied into the image.

As the last part of the reconstruction, an inverse lifting scheme [20] of the Dubuc-Deslauriers-Lemire (4, 4) wavelet transform [19] is carried out. In one dimension, the transform consists of two steps. In the first step, all coefficients c_k with even k are replaced by $c_k - \text{divround}(9(c_{k-1} + c_{k+1}) - (c_{k-3} + c_{k+3}), 32)$, where zero-extension is applied to coefficients lying outside of the image and $\text{divround}(x, y) := \text{floor}(x/y + 1/2)$. In the second step all odd-indexed coefficients c_k are replaced by $c_k + \text{divround}(9(c_{k-1} + c_{k+1}) - (c_{k-3} + c_{k+3}), 16)$. Notice that in the above formula c_{k-1} will always belong to the image. However if only c_{k+3} , or only c_{k-3} , or only c_{k-3} and c_{k+3} , do not belong to the image then the above cubic prediction formula is replaced by a linear prediction $c_k + \text{divround}(c_{k+1} + c_{k-1}, 2)$. In the remaining two cases (c_{k+1} , c_{k+3} do not belong to the image and c_{k-3} does or does not belong to the image) a constant prediction $c_k + \text{divround}(c_{k-1}, 1)$ is employed. The above transform is then carried out on rows and columns interchangeably, mutatis mutandis for 5 wavelet decomposition

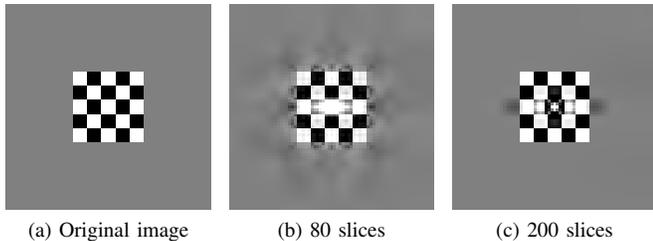


Fig. 4. Critical test image: Size 73×73 with a checker-board pattern with fields of size 5×5 . This image causes an integer overflow in the storage of the wavelet coefficients. Consequently, even the principally loss-less version with 200 slices exhibits artifacts.

levels. After the wavelet transform, the 6 least significant bits of all coefficients are removed by replacing all coefficients c_k with $\text{divround}(c_k, 64)$. The result is clipped to 8-bit signed range $-128, \dots, 127$.

Finally, for both color and grayscale images, the luminance channel Y is shifted by adding 128. For color images a transformation to the RGB space is the carried out and resulting channels are then clipped to $0, \dots, 255$.

Remark. As mentioned in the previous sections, the dynamic range allocated for the wavelet coefficients is 15 bits (with an additional sign bit). The range of the color channel is essentially a 7 bit number (+ 1 additional sign bit). Prior to the wavelet transform, this number is multiplied by 64, hence the input of the wavelet transform is essentially a 13 bit (+1 sign bit). This means that the output of the wavelet transform is only stored correctly if the ∞ norm of the synthesis matrix is at most 4. However, a simple computation shows this not to be the case. Consequently, there are images which are not stored (and therefore not decoded) correctly due to this ‘too small’ dynamic range. An example of this integer overflow/underflow is depicted in Figure 4.

III. VARIATIONAL DJVU RECONSTRUCTION

We can learn from the explanations in the previous section that, after inverting the lossless compression, a DjVu image layer provides a set of DDL $(4, 4)$ wavelet coefficients of the unknown, uncompressed image. As result of the data loss during compression, these coefficients are, however, only known up to a certain level of precision. The DjVu standard decompression accounts for that by estimating the true coefficients to be in the center of their admissible range, which is defined via the level of precision up to which each coefficient is known.

This choice, however, is somewhat arbitrary and our goal is to improve upon that. To this aim, we interpret the admissible range for each coefficient, which can be determined from the compressed file, as given data interval. This then yields a set of admissible wavelet data, henceforth denoted by D , such that the DDL $(4, 4)$ wavelet transform of the uncompressed image is guaranteed to be contained in this set.

Continuing on these lines, the decompression of a given DjVu image layer in fact amounts to choose an image such that the coefficients of its wavelet transform are contained in D . As mentioned above, standard DjVu decompression achieves

this by just setting the first unknown bit to one and all others to zero, which corresponds to choosing the center points of the error intervals.

Our approach is to make a more sophisticated choice which is motivated by a predefined image model. This is achieved in the context of variational regularization for image processing: We regard an image as a function mapping from a rectangular domain to the reals (for grayscale images) or, more generally, to \mathbb{R}^{N_c} (with $N_c = 3$ for color images). An image model is then realized via a regularization functional \mathcal{R} , mapping from the vector space of images to the positive reals, thereby assigning each image to a number which is the lower the better it fits to the underlying model. Consequently, defining W to be the DDL wavelet operator (performing five levels of decomposition), variational DjVu decompression amounts to obtain the reconstructed image as the solution of the minimization problem

$$\min_u \mathcal{R}(u) \quad \text{such that} \quad Wu \in D. \quad (1)$$

This corresponds to choosing, among all possible source images of the compressed DjVu file, the (not necessarily unique) one that best fits the image model.

Naturally, the regularization functional realizing the image model hence plays a crucial role for the quality of the reconstructed image as well as for the computational feasibility of such an approach. Intense research has been devoted to finding appropriate regularization functionals that are both computationally tractable and suitable for a large range of realistic images. The Total Variation (TV) functional [3], which essentially penalizes the one-norm of the image gradient, is a popular choice. Regarding images a functions, a main advantage of TV is that it achieves derivative-based regularization while still allowing jump discontinuities, which in practice yields sharp edges in the reconstructed images. From the computational viewpoint, convexity of TV is another strong advantage as it allows, when combined with convex data constraints, the approximation of provably optimal solutions.

However, a well-known defect of TV regularization is the fact that it tends to produce piecewise constant reconstructions [21], which is infeasible for many realistic images. A straightforward way of overcoming this would for instance be to penalize the one-norm of the second order derivative instead of the first order one [22]. However, again regarding images as functions, this leads a class of possible reconstructions that cannot have jump discontinuities, which in practice corresponds to blurred edges.

The *Total Generalized Variation (TGV)* functional (of order k) [4], [23], allows to overcome the defect of TV while still allowing jump discontinuities. This is achieved in a convex setting via an optimal balancing between different orders of differentiation up to order k . In the second order case, it is defined as

$$\text{TGV}_\alpha^2(u) = \min_v \alpha_1 \|\nabla u - v\|_{\mathcal{M}} + \alpha_0 \|\mathcal{E}v\|_{\mathcal{M}},$$

with $\|\cdot\|_{\mathcal{M}}$ being the Radon norm, i.e., an L^1 -type norm. The balancing is done by subtraction of the vector (or matrix) field v , which in turn is penalized via a symmetrized gradient

$\mathcal{E}v = \frac{1}{2}(\mathcal{J}v + (\mathcal{J}v)^T)$, with $\mathcal{J}v$ being the Jacobian. As v is chosen to minimize the sum of the two penalties, it can be expected to be zero in regions where the image is already constant, i.e., where $\nabla u = 0$, and to be equal to ∇u in regions where the image is almost linear, yielding a penalization of the Hessian $Hu = \mathcal{E}\nabla u$. By choosing $v = 0$ one observes that TGV is always less or equal to TV, hence still admits finite values in case of jump discontinuities.

Motivated by the above discussion, we test our variational DjVu decompression scheme with both TV and second order TGV regularization, reflecting a classical and a state-of-the-art approach, respectively. As the TV regularized framework can be considered as simplification of the TGV setting, we only describe the latter in detail: Images are considered as \mathbb{R}^{N_c} -valued functions ($N_c \in \{1, 3\}$) on a rectangular domain Ω , which we discretize via $\Omega = \{(i, j) \mid 1 \leq i \leq N_x, 1 \leq j \leq N_y\}$. Abusing notation, we carry out identification:

- Vector fields $u : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^{N_c}$ are identified with $u \in U := \mathbb{R}^{N_x \times N_y \times N_c}$,
- matrix fields $v : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^{N_c \times 2}$ are identified with $v \in V := \mathbb{R}^{N_x \times N_y \times N_c \times 2}$,
- tensor fields $w : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^{N_c \times 2 \times 2}$ are identified with $w \in \mathbb{R}^{N_x \times N_y \times N_c \times 2 \times 2}$.

Lower case indices always index space dimensions and upper case indices always index range dimensions, e.g., $u = (u_{i,j}^c)_{i,j,c}$ where $i = 1, \dots, N_x$, $j = 1, \dots, N_y$ and $c = 1, \dots, N_c$. Since the tensor fields are used to represent the symmetrized gradient $\mathcal{E}v$, they will be symmetric around the last two coordinates, i.e.,

$$(w)_{i,j}^c = \begin{bmatrix} w_{i,j}^{c,1,1} & w_{i,j}^{c,1,2} \\ w_{i,j}^{c,1,2} & w_{i,j}^{c,2,2} \end{bmatrix}.$$

Hence, they have a reduced degree of freedom and we identify

- Symmetric tensor fields $w : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}_{\text{sym}}^{N_c \times 2 \times 2}$ with $w \in W := \mathbb{R}^{N_x \times N_y \times N_c \times 3}$.

Again abusing notation, we define pointwise norms $|\cdot|$ on vector and matrix fields as the pointwise Euclidean and Frobenius norms, respectively, e.g., $|u|_{n,m} = \left(\sum_{i=1}^{N_c} (u_{n,m}^i)^2\right)^{1/2}$, and a pointwise norm on symmetric tensor fields as the pointwise Frobenius norm where the coordinated representing the symmetric information is counted twice, i.e.,

$$|w|_{n,m} = \left(\sum_{i=1}^{N_c} (w_{n,m}^{i,1})^2 + (w_{n,m}^{i,2})^2 + 2(w_{n,m}^{i,3})^2\right)^{1/2}.$$

For the approximation of the gradient and symmetrized gradient, we need forward and backward finite differences which, for a vector r of length N , are defined as

$$(\partial^+ r)_i = \begin{cases} r_{i+1} - r_i & \text{for } 1 \leq i \leq N-1 \\ 0 & \text{for } i = N \end{cases}$$

and

$$(\partial^- r)_i = \begin{cases} r_i - r_{i-1} & \text{for } 2 \leq i \leq N \\ 0 & \text{for } i = 1 \end{cases}$$

respectively. For a multi-dimensional array we define the forward and backward differences ∂_x^+ , ∂_y^+ and ∂_x^- , ∂_y^- along

the x and y dimension accordingly, mapping $\mathbb{R}^{N_x \times N_y}$ to itself.

The gradient and symmetrized gradient operators for vector and matrix fields, respectively, are then given as

$$\nabla u = \left(\begin{bmatrix} \partial_x^+ \\ \partial_y^+ \end{bmatrix} [u^1 \quad \dots \quad u^{N_c}] \right)^T$$

and

$$(\mathcal{E}v)^c = \left(\begin{bmatrix} \partial_x^- & 0 \\ 0 & \partial_y^- \\ \frac{1}{2}\partial_x^- & \frac{1}{2}\partial_y^- \end{bmatrix} \begin{bmatrix} v^{c,1} \\ v^{c,2} \end{bmatrix} \right)^T,$$

where the transposition is carried out pointwise in space. Denoting by $\|\cdot\|_1$ the ℓ^1 norm on $\mathbb{R}^{N_x \times N_y}$, the discrete approximation of the TGV functional is then given for $u \in U$ as

$$\text{TGV}_\alpha^2(u) = \min_{v \in V} \alpha_1 \|\nabla u - v\|_1 + \alpha_0 \|\mathcal{E}v\|_1.$$

We further define the discrete DDL wavelet transform operator W (cf. Section II) and, given a DjVu compressed file, denote by D the set of admissible wavelet coefficients. The discrete optimization problem for TGV regularized variational DjVu decompression is then given as

$$\min_{Wu \in D} \text{TGV}_\alpha^2(u). \quad (2)$$

This is a non-smooth but convex optimization problem and, as described in the next section, we will employ duality-based methods to approximate an optimal solution.

Regarding well-posedness, we remark that the minimization problem (2) in fact is a special case of the abstract class of problems considered in [5]. In particular, the existence result [5, Proposition 3.1] guarantees existence of a solution even for a infinite-dimensional version of (2), and in particular we can ensure existence for the discretized problem.

We close this section with a remark regarding the operator wavelet operator W . A close inspection of Subsection II-B shows that the wavelet decomposition as implemented by the lifting scheme is not linear. This is due to the rounding-division function $\text{divround}()$. However, for all intents and purposes the resulting operator can be replaced by a linear operator, where the rounding-division is replaced by the true division, i.e. $\text{divround}(x, y) := x/y$. This is the only concession which we make in modeling the DjVu processing pipeline, for the remainder of the paper. As we will show in the results section this replacement does not have a significant impact on the quality of the reconstructions.

IV. NUMERICAL SOLUTION

It is the goal of the section to derive a globally convergent algorithm that approximates an optimal solution of (2). To this aim, we employ the primal-dual algorithm of [24], which allows to efficiently solve convex-concave saddle-point problems of the form

$$\min_x \max_y (Kx, y) - F^*(y) + G(x),$$

with $F^*(y) := \sup_z (z, y) - F(z)$ being the convex conjugate of a function F , if both F^* and G are prox-simple, meaning that solutions to $\min_u \frac{1}{2}\|u - u_0\|_2^2 + \lambda H(u)$ with $H \in \{F^*, G\}$

can be computed efficiently and fast. Note that, due to strict convexity and coercivity of the 2-norm, given any u_0 , a unique solution to the last equation indeed always exists, and the corresponding solution mapping is called the proximal mapping of H . We will essentially follow the derivation of a JPEG 2000 decompression algorithm in [6]. At first note that TGV can be re-written as

$$\text{TGV}_\alpha^2(u) = \min_v \|K_1(u, v)\|_{1,\alpha}$$

with

$$K_1 = \begin{bmatrix} \nabla & -\text{id} \\ 0 & \mathcal{E} \end{bmatrix}, \quad \|(p, q)\|_{1,\alpha} = \alpha_1 \|p\|_1 + \alpha_0 \|q\|_1.$$

A straightforward computation shows that the convex conjugate of $\|\cdot\|_{1,\alpha}$ is given as

$$\|(p, q)\|_{1,\alpha}^* = \begin{cases} 0 & \text{if } \max\{\alpha_1 \|p\|_\infty, \alpha_0 \|q\|_\infty\} \leq 1, \\ \infty & \text{else.} \end{cases}$$

The data constraint in (2) can be written by adding the convex indicator function $\mathcal{I}_D(Wu)$ to the objective function, where $\mathcal{I}_D(r) = 0$ if $r \in D$ and $\mathcal{I}_D(r) = \infty$ else.

The convex conjugate of \mathcal{I}_D is then given as $\mathcal{I}_D^*(r) = \sup_{s \in D} (r, s)$. The reformulation of (2) is now carried out as follows:

$$\begin{aligned} & \min_{Wu \in D} \text{TGV}_\alpha^2(u) \Leftrightarrow \min_{u,v} \|K_1(u, v)\|_{1,\alpha} + \mathcal{I}_D(Wu) \\ \Leftrightarrow & \min_{u,v} \max_{p,q,r} (K_1(u, v), (p, q)) - \|(p, q)\|_{1,\alpha}^* + (Wu, r) - \mathcal{I}_D^*(r) \\ \Leftrightarrow & \min_{u,v} \max_{p,q,r} (K(u, v), (p, q, r)) - F^*(p, q, r), \end{aligned}$$

with

$$K = \begin{bmatrix} \nabla & -\text{id} \\ 0 & \mathcal{E} \\ W & 0 \end{bmatrix} \quad \text{and} \quad F^*(p, q, r) = \|(p, q)\|_{1,\alpha}^* + \mathcal{I}_D^*(r)$$

the convex conjugate of $F(p, q, r) = \|(p, q)\|_{1,\alpha} + \mathcal{I}_D(r)$. The last line of the above equivalence defines a saddlepoint problem that can be solved by the algorithm of [24] and, as can be deduced with standard convex analysis arguments [25], is equivalent to the original problem in the sense that, given $((u, v), (p, q, r))$ a solution to the saddle-point problem, u is an optimal solution of (2). The resulting algorithm is provided in Algorithm 1. There, the operators proj_{α_1} , proj_{α_0} result from the proximal mapping of $\|\cdot\|_{1,\alpha}^*$ and can be calculated explicitly as

$$(\text{proj}_\lambda(s))_{n,m}^{c,i} = s_{n,m}^{c,i} / \max\{1, |s_{i,j}|/\lambda\}.$$

The operator $\text{shrink}_{D,\sigma}$ results from the proximal mapping of \mathcal{I}_D^* and can be calculated explicitly as

$$(\text{shrink}_{D,\sigma}(r))_{n,m}^c = \begin{cases} r_{n,m}^c - \sigma o_{n,m}^c & \text{if } r_{n,m}^c > \sigma o_{n,m}^c \\ r_{n,m}^c - \sigma l_{n,m}^c & \text{if } r_{n,m}^c < \sigma l_{n,m}^c \\ 0 & \text{else.} \end{cases}$$

where $J_{n,m}^c = [l_{n,m}^c, o_{n,m}^c]$ defines the data constraint of the wavelet coefficient (n, m) of the color component c for the data set D . In contrast to the original implementation as provided in [24], we use an adaptive choice of the stepsizes

Algorithm 1 Algorithm for variational DjVu decompression

```

1: function TGV-DjVu(FILE)
2:    $D \leftarrow$  interval bounds from decoding of FILE
3:    $d \leftarrow$  centers of interval bounds,  $u \leftarrow W^{-1}(d)$ 
4:    $v \leftarrow 0, \bar{u} \leftarrow u, \bar{v} \leftarrow 0, p \leftarrow 0, q \leftarrow 0, r \leftarrow 0$ 
5:   choose  $\sigma, \tau > 0, \eta > 0$ 
6:   repeat
7:      $p \leftarrow \text{proj}_{\alpha_1}(p + \sigma(\nabla \bar{u} - \bar{v}))$ 
8:      $q \leftarrow \text{proj}_{\alpha_0}(q + \sigma \mathcal{E} \bar{v})$ 
9:      $r \leftarrow \text{shrink}_{D,\sigma}(r + \sigma W \bar{u})$ 
10:     $u_+ \leftarrow u - \tau(-\text{div } p + W^* r)$ 
11:     $v_+ \leftarrow v - \tau(-p - \text{div}_2 q)$ 
12:     $\bar{u} \leftarrow (2u_+ - u), \bar{v} \leftarrow (2v_+ - v)$ 
13:     $\sigma_+ \leftarrow \eta \mathcal{S}(\sigma \tau, \frac{\|(u_+, v_+) - (u, v)\|_2}{\|K((u_+, v_+) - (u, v))\|_2})$ 
14:     $\tau_+ \leftarrow \sigma_+ / \eta^2$ 
15:     $u \leftarrow u_+, v \leftarrow v_+$ 
16:  until Stopping criterion fulfilled
17:  return  $u_+$ 
18: end function

```

σ and τ according to [6], which still ensures convergence of the algorithm while in practice allowing to overcome stepsize constraint $\sigma \tau \|K\|^2 < 1$ provided in [24]. The adaptive choice is realized via the mapping \mathcal{S} , which is defined as

$$\mathcal{S}(\sigma \tau, \mathbf{n}) = \begin{cases} \mathbf{n} & \text{if } \sqrt{\theta \sigma \tau} \geq \mathbf{n}, \\ \sqrt{\theta \sigma \tau} & \text{if } \sqrt{\sigma \tau} \geq \mathbf{n} > \sqrt{\theta \sigma \tau}, \\ \sqrt{\sigma \tau} & \text{else.} \end{cases} \quad (3)$$

The parameter η defines the ratio between the primal and the dual step and was set to $\eta = 0.001$ for all experiments, as this accelerated convergence. The norm $\|\cdot\|_2$ in Algorithm 1 refers to the standard ℓ^2 norm over all pixels and the pointwise norm $|\cdot|$ is meant component-wise. The operators $\text{div} : V \rightarrow U$ and $\text{div}_2 : W \rightarrow V$ are defined as the negative adjoints of ∇ and \mathcal{E} , respectively, with respect to the standard inner product. As stopping criterion, we use the normalized primal dual gap defined in [6], which is for our setting given as

$$\begin{aligned} \mathcal{G}(u, v, p, q, r) = & \frac{1}{N_x N_y N_c} \left(F(K(u, v)) + \mathcal{D}(Wu, r) \right. \\ & \left. + \gamma \|u\|_2 \|\text{div}_2 \tilde{q} - W^* r\|_2 \right) \\ & + \sum_{(i,j,c)} \frac{l_{i,j}^c + o_{i,j}^c}{2} (r)_{i,j}^c + \left| \frac{o_{i,j}^c - l_{i,j}^c}{2} (r)_{i,j}^c \right| \end{aligned} \quad (4)$$

where

$$\mathcal{D}(s, r) = \sum_{s_{i,j}^c \notin J_{i,j}^c} \gamma |(r)_{i,j}^c| \max\{s_{i,j}^c - o_{i,j}^c, l_{i,j}^c - s_{i,j}^c\},$$

$\tilde{q}_n = \beta_n q_n$, with $\beta_n := \frac{\alpha_1}{\max(\alpha_1, \|\text{div } q_n\|_\infty)}$ and $\gamma > 1$. Summarizing the variables at iteration n by $x_n = (u_n, v_n)$ and $y_n = (p_n, q_n, r_n)$, it was shown in [6] that, $\mathcal{G}(x_n, y_n) \rightarrow 0$ as $n \rightarrow \infty$ and, after finitely many iterates,

$$0 \leq F(Kx_n) + \mathcal{D}(Wu_n, r_n) - \text{TGV}_\alpha^2(\hat{u}) \leq \mathcal{G}(x_n, y_n),$$

with \hat{u} being an optimal solution to the variational decompression problem (1). In particular, $\mathcal{G}(x_n, y_n) = 0$ if and only if (x_n, y_n) solves the saddle-point problem equivalent to (2).

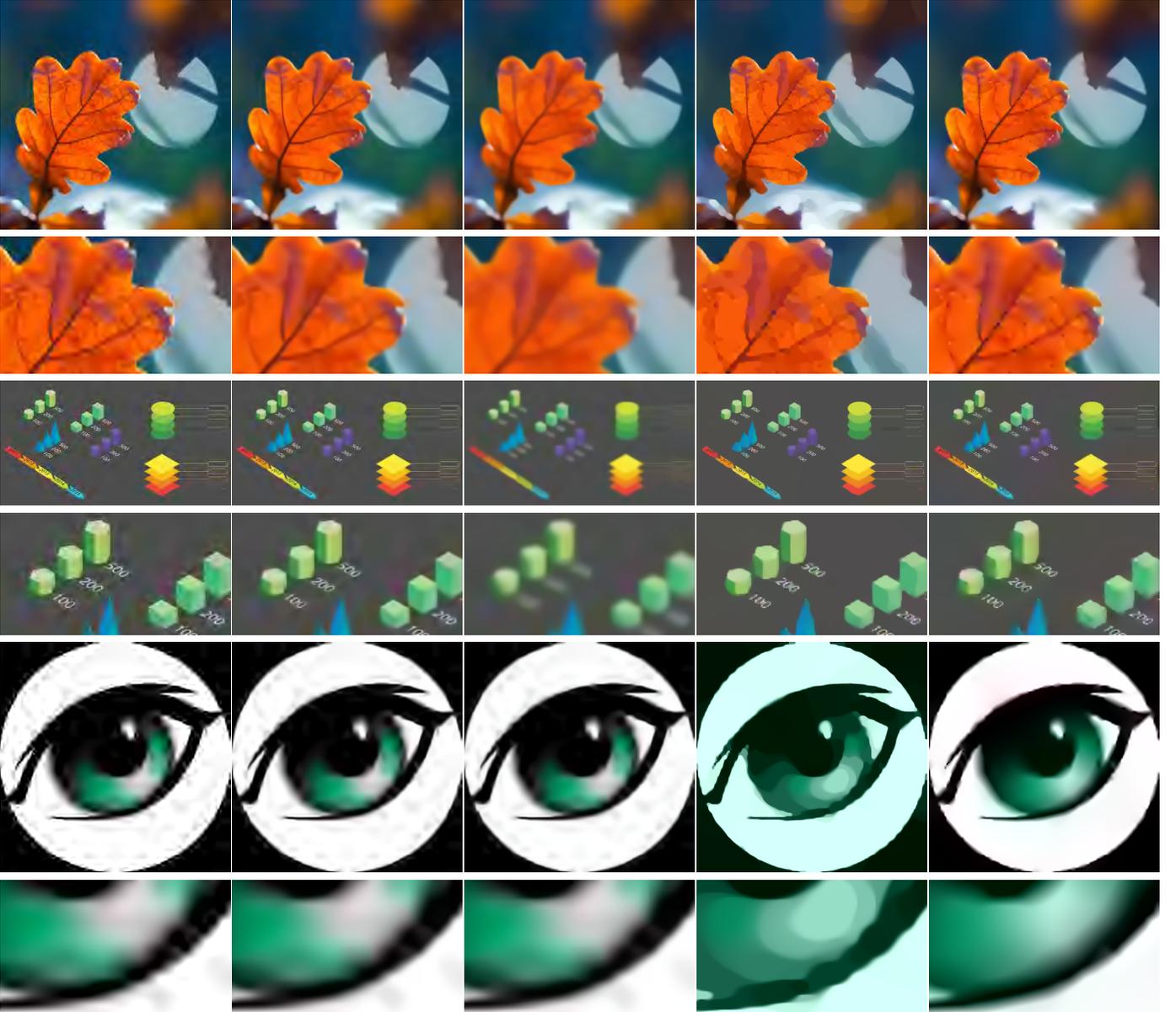


Fig. 5. Comparison of variational decomposition and different postprocessing techniques. From left to right: Standard, BM3D - optimal parameter choice (BM3D-opt), BM3D - parameter choice for strong smoothing (BM3D-max), TV, TGV. [PSNR/SSIM]: Leaf: Standard [28.9/0.91], BM3D-opt [29.4/0.91], BM3D-max [27.8/0.89], TV [29/0.91], TGV [29.1/0.92], Infographics: Standard [27.4/0.92], BM3D-opt [27.9/0.94], BM3D-max [25.4/0.86], TV [29.3/0.96], TGV [28.3/0.95], Eye-clipart: Standard [20.7/0.73], BM3D-opt [21/0.76], BM3D-max [21/0.78], TV [19.1/0.58], TGV [20.9/0.76]. Top image “Leaf” [26] licenced under CC BY-SA 2.0 (<https://creativecommons.org/licenses/by-sa/2.0/>). Middle image “Infographics” and bottom clipart “Eye-clipart” licenced under CC0 (<https://creativecommons.org/about/cc0>).

V. NUMERICAL EVALUATION

The purpose of this section is to evaluate variational DjVu decomposition using TV and TGV regularization numerically and to compare with existing techniques. As, to the best knowledge of the authors, there are no methods available that are particularly tailored towards DjVu image decomposition, we compare to custom modifications of related approaches as well as state-of-the-art denoising methods. Regarding the latter, we stress that the meaningfulness of such a comparison is naturally limited to assessing the extend to which these methods can be “misused” for postprocessing, as they are originally designed for Gaussian noise removal and no

decompression-specific modifications were implemented. The following techniques were used for comparison:

Standard decomposition: The standard decomposition is the result provided by the reference implementation [17].

TV-based decomposition: In addition to the TGV image model, we also test the total variation (TV) model. To this aim, the TGV_{α}^2 functional in (2) is replaced by $TV(u) = \|\nabla u\|_1$. For numerical solution, Algorithm 1 was adapted accordingly.

Block-matching and 3D filtering (BM3D): BM3D [27] is patch-based and filtering is carried on matching patches using both spatial smoothing via appropriate 2D transforms and a smoothing across transformed patches by a thresholding

of Haar wavelet coefficients. For our experiments we have employed the (closed source) implementation provided by the original authors of BM3D [28].

Non-local total variation (NLTV): The non-local total variation functional $NLTV(u) = \sum_i \left(\sum_j (u_i - u_j)^2 w_{i,j} \right)^{1/2}$, as proposed in [29], was employed in [8] for wavelet inpainting. Translated to the setting of this paper, this comprises the solution of $\min_{W} \min_{u \in D} NLTV(u)$, where the set D is a product-set of sets which are singletons or the whole set of real numbers \mathbb{R} . In the light of image decomposition this can be interpreted as either knowing the related wavelet-coefficients exactly (singleton case) or not at all (whole \mathbb{R} case). Simulating this situation, we provide a comparison of our method to NLTV regularization using the reference implementation provided by one of the authors [30], which uses an iterative update of the weights $(w_{i,j})_{i,j}$.

For NLTV, standard parameters as given in the reference implementations were employed. The main parameter in BM3D is the estimated noise-level which corresponds to the amount of smoothing. For our evaluation, we tested a range of different parameters and present results with both the optimal parameter choice (w.r.t. PSNR) and a parameter choice corresponding to strong smoothing ($\sigma = 500$). The TV- and TGV-based variational methods are essentially parameter free: TGV requires only to fix the ratio $\frac{\alpha_0}{\alpha_1}$, determining the trade-off between first and second order derivative information, however, a fixed choice of $\sqrt{2}$ can be used and is known to work reasonably well across different imaging applications [5], [31].

A crucial a-priori information for the proposed algorithm is the knowledge of the admissible intervals for the wavelet-coefficients, cf. D in (2). This is virtually equivalent to a decoding of the bitstream information of a DjVu-file. We have implemented such decoding routine in MATLAB, relying of the DjVu standard in version 2 and 3. For certain details of the implementation of the ZP-codec we have relied on the reference implementation by djvulibre library, cf. [17]. To provide a general basis for variational DjVu decomposition as well as for the sake of reproducibility, source code implementing all data-related aspects of DjVu decomposition as well as the TV and TGV reconstruction code are available in the supplementary material of this paper.

A first example of the effect of our method on the DjVu page used in the introduction is shown in Figure 1. There, both the fore- and background layer was compressed and afterwards reconstructed by the TGV-based method. The results show an improvement of image quality compared to standard decomposition, both for the photographic part and the pie chart.

In Figure 5 we provide a qualitative evaluation of the framework on three images comparing standard as well as TV- and TGV-based DjVu decomposition with postprocessing by BM3D denoising. Further experiments with non-local means denoising [32] and additional images can be found in the supplementary material of this paper. The TV- and TGV-based reconstruction was obtained using $\mathcal{G}(u, v, p, q, r) < 0.2$, with \mathcal{G} as in (4), as stopping criterion. One photographic and two synthetic images are presented. The compression ratio in DjVu is defined via the number of slices (see Section



Fig. 6. Top: Standard decomposition of DjVu file, bottom: TGV-based decomposition of same file. Left: Entire page, right: Close up-views.

II) and we use 75 slices for the photo, 80 slices for the *Infographics* image and 50 slices for the *Eye-clipart*. Visual inspection shows that in particular the TGV-based method allows to significantly improve upon standard decomposition. This is especially evident in the round reflection part of the “Leaf” example as well as in the interfaces of the “Eye-clipart” example. For the “Infographics” example both TV and TGV strongly improve upon standard decomposition. A postprocessing with BM3D filtering also improves image quality, but with the optimal parameter choice some artifacts remain while a high parameter value leads to oversmoothing. In contrast to that, the TV-based variant is able to remove compression artifacts and keeps edges sharp, however, at the cost of degrading image quality by introducing regions with piecewise constant values (the staircasing effect). In addition, a green-shift can be observed with TV for the clip-art image, which can be explained by the fact that at this compression level almost no data constraints for the color components are left and TV hence pushes them to be almost constant. Enforcing piecewise smoothness, as done with the TGV-based method, does not suffer from this artifact and leads to sharp edges with natural transitions in-between. For cartoon-like images, such as the “Eye-clipart”, this allows a very good reconstruction at a high compression level. Due to the hard wavelet constraints obtained from the compressed file, small details in the photographic images are not over-smoothed.

These qualitative differences in the reconstruction, however, are not well captured by the standard image similarity measures peak signal-to-noise ratio (PSNR) and structure similarity index (SSIM) [33], whose values are very similar and do not reflect the visual quality of the reconstructions.

At last, Figure 7 compares different regularization approaches for improving upon the n-term approximation of a given image using 5% of the coefficients of the five-level



Fig. 7. Simulated decompression with n -term wavelet approximation (5% of coefficients). From Left to right: Inverse DWT, NLTV, TGV. [PSNR/SSIM]: Standard [29.6/0.87], NLTV [27.3/0.85], TGV [29.3/0.88]

DDL (4, 4) wavelet transform. To account for the larger set of admissible images, resulting from unbounded data intervals with the n -term approximation, the slightly more restrictive stopping criterion $\mathcal{G}(u, v, p, q, r) < 0.1$, again with \mathcal{G} as in (4), was used for the TGV reconstructions. The results are presented in Figure 7. As in the case of color image decompression, the presented method is able to reliably recover sharp edges and smooth transitions. Wavelet artifacts with the zero-fill wavelet reconstruction are more severe this time, and consequently, the visual improvement with the TGV-based method is even stronger. Again, however, quantitative evaluation gives comparable results indicating a weak relation of error measures and visual image quality. The NLTV-based method also removes wavelet artifacts, but introduces new artifacts, such as around the parrots eye. This might be explained by the fact that NLTV method uses an iterative update of the weights, a procedure which makes the overall problem non-convex.

VI. CONCLUSION

In this paper, a framework for the variational decompression of image data from DjVu files is presented. Its main contribution is to establish both the technical and analytical foundations for variational approaches to DjVu decompression in general. This includes a detailed description of all necessary steps within the DjVu encoding/decoding pipeline, a derivation of appropriate (convex) data constraints together with a corresponding regularization model as well as supplementary software tools.

Additionally, the paper describes the concrete numerical realization of variational DjVu decompression using state of the art regularization approaches. A numerical evaluation further indicates the potential of variational decompression. In particular, it shows that TGV-based decompression is able to remove strong compression artifacts without over-smoothing the image.

REFERENCES

[1] O. Scherzer, M. Grasmair, H. Grossauer, M. Haltmeier, and F. Lenzen, *Variational Methods in Imaging*. Springer, 2008.

[2] P. Haffner, P. G. Howard, P. Simard, Y. Bengio, Y. Lecun *et al.*, “High quality document image compression with DjVu,” *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410–425, 1998.

[3] L. Rudin, S. Osher, and E. Fatemi, “Nonlinear Total Variation based noise removal algorithms,” *Phys. D*, vol. 60, pp. 259–268, 1992.

[4] K. Bredies, K. Kunisch, and T. Pock, “Total Generalized Variation,” *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 492–526, 2010.

[5] K. Bredies and M. Holler, “A TGV-based framework for variational image decompression, zooming and reconstruction. Part I: Analytics,” *SIAM J. Imaging Sci.*, vol. 8, pp. 2214–2850, 2015.

[6] —, “A TGV-based framework for variational image decompression, zooming and reconstruction. Part II: Numerics,” *SIAM J. Imaging Sci.*, vol. 8, pp. 2851–2886, 2015.

[7] Y.-W. Wen, R. H. Chan, and A. M. Yip, “A primal-dual method for total variation-based wavelet domain inpainting,” *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 106–114, 2011.

[8] X. Zhang and T. F. Chan, “Wavelet inpainting by nonlocal Total Variation,” *Inverse Probl. Imag.*, vol. 4, no. 1, pp. 191–210, 2010.

[9] T. F. Chan, J. Shen, and H.-M. Zhou, “Total Variation wavelet inpainting,” *J. Math. Imaging Vision*, vol. 25, no. 1, pp. 107–125, 2006.

[10] S. Zhong, “Image coding with optimal reconstruction,” in *ICIP, IEEE*, vol. 1, 1997, pp. 161–164.

[11] F. Alter, S. Durand, and J. Froment, “Adapted Total Variation for artifact free decompression of jpeg images,” *J. Math. Imaging Vision*, vol. 23, no. 2, pp. 199–211, 2005.

[12] K. Bredies and M. Holler, “A Total Variation-based JPEG decompression model,” *SIAM J. Imaging Sci.*, vol. 5, no. 1, pp. 366–393, 2012.

[13] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, and F. Wu, “A highly parallel framework for hevcc coding unit partitioning tree decision on many-core processors,” *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573–576, 2014.

[14] C. Yan, Y. Zhang, J. Xu, F. Dai, J. Zhang, Q. Dai, and F. Wu, “Efficient parallel framework for hevcc motion estimation on many-core processors,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2077–2089, 2014.

[15] [Online]. Available: <http://sourceforge.net/projects/djvu/files/DjVuLibre/>

[16] [Online]. Available: <http://www.djvuzone.org>

[17] [Online]. Available: <http://djvu.sourceforge.net/>

[18] J. Morrison, “Iff standard,” 1985. [Online]. Available: <http://www.digitalpreservation.gov/formats/fdd/fdd000115.shtml>

[19] G. Deslauriers, S. Dubuc, and D. Lemire, “Une famille d’ondelettes biorthogonales sur l’intervalle obtenue par un schéma d’interpolation itérative,” *Ann. Sci. Math. Québec*, vol. 23, pp. 37–48, 1999.

[20] W. Sweldens, “The lifting scheme: A custom-design construction of biorthogonal wavelets,” *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, 1996.

[21] W. Ring, “Structural properties of solutions to Total Variation regularization problems,” *Esaim Math Model Numer Anal*, vol. 34, pp. 799–810, 2000.

[22] W. Hinterberger and O. Scherzer, “Variational methods on the space of functions of bounded hessian for convexification and denoising,” *Computing*, vol. 76, no. 1, pp. 109–133, 2006.

[23] K. Bredies and M. Holler, “Regularization of linear inverse problems with Total Generalized Variation,” *J. Inverse Ill-Posed Probl.*, vol. 22, no. 6, pp. 871–913, 2014.

[24] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” *J. Math. Imaging Vision*, vol. 40, pp. 120–145, 2011.

[25] I. Ekeland and R. Témam, *Convex Analysis and Variational Problems*. SIAM, 1999.

[26] Steve-h, “Oak leaves and bokeh.” [Online]. Available: <https://www.flickr.com/photos/sbh/6802942537/>

[27] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-d transform-domain collaborative filtering,” *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.

[28] [Online]. Available: <http://www.cs.tut.fi/foi/GCF-BM3D/>

[29] G. Gilboa and S. Osher, “Nonlocal operators with applications to image processing,” *Multiscale Model Simul*, vol. 7, no. 3, pp. 1005–1028, 2008.

[30] [Online]. Available: <http://math.sjtu.edu.cn/faculty/xqzhang/>

[31] F. Knoll, K. Bredies, T. Pock, and R. Stollberger, “Second order Total Generalized Variation (TGV) for MRI,” *Magn. Reson. Med.*, vol. 65, no. 2, pp. 480–491, 2011.

[32] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” in *CVPR*, vol. 2. IEEE, 2005, pp. 60–65.

[33] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.