IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2013–07–12.
Submitted on 2012–09–04, accepted on 2013–02–09.
ISSN 2105–1232 © 2013 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2013.40

# Combined First and Second Order Total Variation Inpainting using Split Bregman

Konstantinos Papafitsoros[1], Carola-Bibiane Schönlieb[2], Bati Sengul[3]

[1] Cambridge Centre for Analysis, University of Cambridge, UK (k.papafitsoros@maths.cam.ac.uk)
[2] Cambridge Centre for Analysis, University of Cambridge, UK (C.B.Schoenlieb@damtp.cam.ac.uk)
[3] Cambridge Centre for Analysis, University of Cambridge, UK (b.sengul@maths.cam.ac.uk)

**Abstract**

In this article we discuss the implementation of the combined first and second order total variation inpainting that was introduced by Papafitsoros and Schönlieb. We describe the algorithm we use (split Bregman) in detail, and we give some examples that indicate the difference between pure first and pure second order total variation inpainting.

**Source Code**

We provide a source code for the algorithm written in C and an online demonstration, accessible on the article web page http://dx.doi.org/10.5201/ipol.2013.40.

**Keywords:** inpainting, split Bregman, second order total variation

## 1 Introduction – TV-TV$^2$ Image Inpainting

Image inpainting denotes the task of restoring a missing part of an image using information from the known part. In the continuous setting, a greyscale image is typically modelled as a function $u : \Omega \to \mathbb{R}$ where $\Omega \subseteq \mathbb{R}^2$ is usually a rectangle and $u(x)$ is the intensity of the grey level at the point $x$. The missing part of the image is called the *inpainting domain* and is denoted by $D \subseteq \Omega$, see figure 1. In the discrete setting an image is a function $u : \{1, 2, \dots, n\} \times \{1, 2, \dots, m\} \to \mathbb{R}$, where $n \times m$ is the image resolution ($nm$ pixels). Coloured images are modelled as vector valued functions, i.e., with values in $\mathbb{R}^3$ instead of $\mathbb{R}$. That is, one real value per colour channel, e.g. for the red, green and blue channels (RGB).

Image inpainting methods can be roughly separated into four categories, depending on being *variational* or *non-variational* and *local* or *non-local*. The variational methods in contrast with the non-variational are characterised by the fact that the reconstructed image $u_r$ is obtained as a minimiser of a certain energy functional. A method is local if the information that is needed to fill in the inpainting domain is only taken from points neighboring the boundary of $D$. Non-local or global inpainting methods take into account all the information from the known part of the image, usually weighted by its distance to the point that is to be filled in. The latter class of methods is

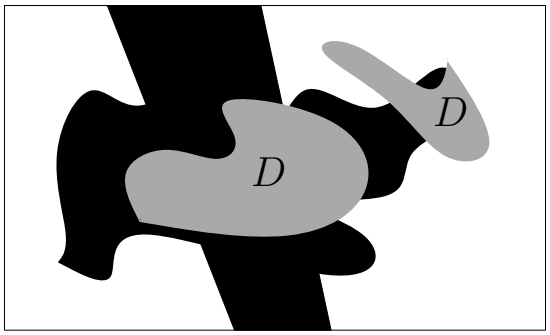



Figure 1: The inpainting domain $D$ is marked with grey colour. The task is to fill in this domain in a sensible way using information from the rest of the image.

(a) Original image.


(b) Image with text.

(c) Restored image using TV inpainting with $\alpha = 0.01$. The inpainting domain was chosen to be all the white pixels.

(d) Absolute difference between the original and the restored image - rescaled for better visibility. The PSNR inside the inpainting domain is (29.47, 29.51, 31.54) for the RGB channels respectively.

Figure 2: Removing text from an image using TV inpainting. Note that the absolute error is larger in the areas where the original image has texture. This is because TV inpainting promotes piecewise constant reconstructions inside the inpainting domain.

very powerful, allowing to fill in structures and textures almost equally well. However, they still have some disadvantages. One major one is the high computational cost involved in their solution. Local methods are sometimes more desirable, especially when the inpainting domain is relatively small. Here we discuss only variational, local methods. For different types of local inpainting methods we refer the reader for instance to Bertalmío et al. [3] and Bornemann and März [5] (inpainting via transport), Chan and Shen [10] (TV inpainting), Chan and Shen [9] (curvature driven diffusion inpainting), Esedoglu and Shen [14] (Mumford–Shah based inpainting) and Masnou and Morel [18], Chan et al. [8], and Tai et al. [26] (Euler's elastica inpainting). For non-local methods the reader is referred to Cao et al. [7], Criminisi et al. [12], and Arias et al. [2].

In the variational approach, the central role is held by a term $\Psi(u)$ known as the *regularising term* or *regulariser*. The minimisation of $\Psi(u)$ is responsible for the filling-in process inside the inpainting domain. Of course, different choices of $\Psi$ result in different qualitative restorations of the missing part $D$. In our formulation, a second term $\mathcal{D}_{\Omega\setminus D}(u, u_0)$ is added to $\Psi$, which is nonnegative and has the property that it is zero when $u(x) = u_0(x)$ for every $x \in \Omega \setminus D$. This term is called the *fidelity term*. Since we desire the values of $u$ to be close to $u_0$ in $\Omega \setminus D$ we would like to keep the value of the fidelity term as small as possible. Typically, the fidelity term is chosen to be the square of the $L^2$ norm of $u - u_0$, since this is convex and differentiable. The minimisation problem then reads

$$\min_u \int_{\Omega\setminus D} (u - u_0)^2 \, dx + \Psi(u), \tag{1}$$

i.e., the resulting energy functional is the sum of the fidelity and the regularising term. The fidelity and regularising terms are balanced against each other by one or more regularisation parameters introduced in $\Psi$.

In the continuous setting, the minimisation (1) is done over an appropriate Banach space $\mathcal{B}$ which depends on the choice of the regulariser $\Psi$. In the case that $\Psi(u) = \alpha \int_\Omega |\nabla u|^2 \, dx$ (harmonic inpainting), we have $\mathcal{B} = W^{1,2}(\Omega)$, while in the case that $\Psi(u) = \alpha \mathrm{TV}(u)$, the total variation of $u$ (TV inpainting), we have $\mathcal{B} = \mathrm{BV}(\Omega)$, the space of functions of bounded variation see Ambrosio et al. [1] and Giusti [15]. Here $\alpha$ is a positive parameter balancing the fidelity and the regularising term. For regular enough functions the total variation of $u$ is the $L^1$ norm of the gradient, i.e., $\mathrm{TV}(u) = \int_\Omega |\nabla u| \, dx$. The advantage of TV over harmonic inpainting is that it is able to preserve discontinuities inside the inpainting domain, see Chan and Shen [25, 11]. One of the many explanations for this effect is that by minimising the $L^1$ norm of the gradient, one obtains solutions that have sparse gradient, i.e., piecewise constant images inside the inpainting domain.

In our method, the regularising term is chosen to be the sum of the total variation $\mathrm{TV}(u)$ plus the total variation of the gradient, $\mathrm{TV}^2(u) = \mathrm{TV}(\nabla u)$, weighted with two positive parameters $\alpha$ and $\beta$ respectively, i.e.,

$$\Psi(u) = \alpha \mathrm{TV}(u) + \beta \mathrm{TV}^2(u). \tag{2}$$

The appropriate Banach space for the minimisation (1) is the space of functions of bounded Hessian $\mathrm{BH}(\Omega)$, see Demengel [13]. Hence, the variational problem is

$$u_r = \operatorname*{arg\,min}_{u \in \mathrm{BH}(\Omega)} \int_{\Omega\setminus D} (u - u_0)^2 \, dx + \alpha \mathrm{TV}(u) + \beta \mathrm{TV}^2(u). \tag{3}$$

Here, for sufficiently regular functions $u$, $\mathrm{TV}^2(u)$ is the $L^1$ norm of the Hessian, i.e., $\mathrm{TV}^2 = \int_\Omega |\nabla^2 u| \, dx$. The motivation for introducing this second order term in the regulariser (2) is that, as it was observed by Papafitsoros and Schönlieb [20], this kind of regularisation has the ability to connect large gaps in the inpainting domain with the price of some blur. If the inpainting domain is thin enough, as it is for example in the case of text removal, TV inpainting performs satisfactorily, see figure 2. However, in the task of interpolating large gaps along the inpainting domain, TV and

(a) Small gap. Inpainting domain: grey area.

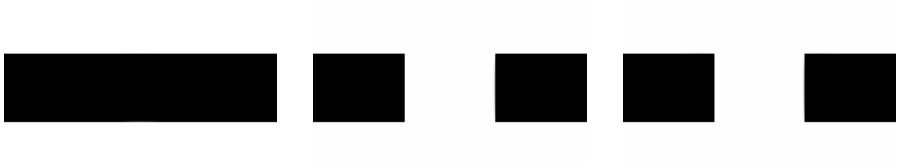
(b) Large gap 1. Inpainting domain: grey area

(c) Large gap 2. Inpainting domain: grey area

(d) Small gap: result of harmonic inpainting with $\alpha = 0.01$.

(e) Large gap 1: result of harmonic inpainting with $\alpha = 0.01$.

(f) Large gap 2: result of harmonic inpainting with $\alpha = 0.01$.



(g) Small gap: result of TV inpainting with $\alpha = 0.01$.

(h) Large gap 1: result of TV inpainting with $\alpha = 0.01$.

(i) Large gap 2: result of TV inpainting with $\alpha = 0.01$.

(j) Small gap: result of pure $TV^2$ inpainting with $\beta = 0.01$.

(k) Large gap 1: result of pure $TV^2$ inpainting with $\beta = 0.01$.

(l) Large gap 2: result of pure $TV^2$ inpainting with $\beta = 0.01$.

Figure 3: (a)–(c): Three different inpainting domains. (d)–(f): Harmonic inpainting. (g)–(i): TV inpainting. (j)–(l): $TV^2$ inpainting.

harmonic inpainting perform poorly in comparison with $\mathrm{TV}^2$ inpainting, see figure 3. In the example in figure 3 the task is to inpaint gaps of different widths in a black stripe. Harmonic inpainting is achieving no connectedness, producing a rather smooth result. TV inpainting is able to connect and preserve the edge of the stripe only if the gap is small enough [8], while $\mathrm{TV}^2$ inpainting is able to connect the large gap with the price of some blur. However, this connectedness depends also on the size and geometry of the inpainting domain, see section 3.2 for some examples and heuristics.

The role of the parameters $\alpha$ and $\beta$ in the combined TV-$\mathrm{TV}^2$ inpainting is twofold. On one hand, one wants to keep both values of $\alpha$ and $\beta$ small such that more weight is put on the fidelity term so it remains close to zero and hence we essentially have $u_r = u_0$ in $\Omega \setminus D$. On the other hand, the values of $\alpha$ and $\beta$ are also balancing the two regularising terms TV and $\mathrm{TV}^2$. Although, as we see in section 3.1, connectedness along large gaps is essentially achieved with pure $\mathrm{TV}^2$ inpainting ($\alpha = 0$), we introduce this combination in order to keep the method flexible, regarded as a superset of pure TV and pure $\mathrm{TV}^2$ inpainting. In this way, we can study the effect that each term has on the inpainting image.

For the well-posedness of minimisation (1), i.e., existence of minimisers as well as stability results, we refer the reader to the respective references for the models we mentioned above. In particular, the well-posedness of the minimisation problem (3) is proven in Papafitsoros and Schönlieb [20]. Even though, existence and stability results are proven, uniqueness of solutions is most of the time absent due to the non-strict convexity of the fidelity term and the regulariser. This non-uniqueness is not a fault of the variational problem but is correctly reflecting the ambiguity of the inpainting problem.

For more information on variational inpainting involving higher order derivatives, see e.g. Chan et al. [8], Lai et al. [17], Chan and Shen [11], Masnou and Morel [18] (Euler's elastica), Esedoglu and Shen [14] (Mumford–Shah–Euler inpainting), Bertozzi et al. [4], Schönlieb and Bertozzi [24] (low curvature image simplifiers).

## 2 Implementation using the Split Bregman Method

In this section we describe in detail how we solve numerically the minimisation problem (3). We use the split Bregman method as it was done in Papafitsoros and Schönlieb [20]. We start by formulating the discrete version of the problem (3). We define the corresponding $L^2$ and $L^1$ norms of a discrete vector field $U : \{1, 2, \dots, n\} \times \{1, 2, \dots, m\} \to \mathbb{R}^d$ with $U = (U_1, U_2, \dots, U_d)$ as follows:

$$\|U\|_2 = \left( \sum_{i=1}^{n} \sum_{i=1}^{m} \sum_{k=1}^{d} U_k(i,j)^2 \right)^{1/2},$$

$$\|U\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} \left( \sum_{k=1}^{d} U_k(i,j)^2 \right)^{1/2}.$$

In the discretised setting the minimisation problem we need to solve is the following:

$$\min_{u \in \mathbb{R}^{n \times m}} \|\mathcal{X}_{\Omega \setminus D}(u - u_0)\|_2^2 + \alpha \|\nabla u\|_1 + \beta \|\nabla^2 u\|_1, \tag{4}$$

where $\nabla u = (D_x u, D_y u)$ and $\nabla^2 u = (D_{xx} u, D_{yy} u, D_{xy} u, D_{yx} u)$ are discrete differential operators of first and second order respectively. In the next section, we define the discrete differential operators such that $D_{xy} = D_{yx}$.

## 2.1 Discrete First and Second Order Derivatives

We define here the discrete first and second derivatives $D_x$, $D_y$, $D_{xx}$, $D_{yy}$ and $D_{xy}$ which are all operators from $\mathbb{R}^{n\times m}$ to $\mathbb{R}^{n\times m}$. We assume periodic boundary conditions for $u$. By choosing periodic boundary conditions, the action of each of the discrete differential operators can be regarded as a circular convolution of $u$ and allows the use of fast Fourier transform, which our algorithm relies on, see also Wu and Tai [27] for a similar implementation.

In our discrete implementation, the coordinates $x$ and $y$ are oriented along columns and rows respectively. We define the discrete gradient $\nabla u = (D_x u, D_y u)$ as a forward difference operator

$$D_x u(i,j) = \begin{cases} u(i,j+1) - u(i,j) & \text{if } 1 \le i \le n,\ 1 \le j < m, \\ u(i,1) - u(i,j) & \text{if } 1 \le i \le n,\ j = m, \end{cases}$$

$$D_y u(i,j) = \begin{cases} u(i+1,j) - u(i,j) & \text{if } 1 \le i < n,\ 1 \le j \le m, \\ u(1,j) - u(i,j) & \text{if } i = n,\ 1 \le j \le m. \end{cases}$$

We also need the discrete divergence operator $\mathrm{div} : (\mathbb{R}^{n\times m})^2 \to \mathbb{R}^{n\times m}$ that has the adjointness property

$$-\mathrm{div} p \cdot u = p \cdot \nabla u, \quad \forall u \in \mathbb{R}^{n\times m},\ p \in \left(\mathbb{R}^{n\times m}\right)^2.$$

This property forms the discrete analogue of integration by parts. For a $p = (p_1, p_2) \in (\mathbb{R}^{n\times m})^2$ we define

$$(\mathrm{div} p)(i,j) = \overleftarrow{D_x} p_1(i,j) + \overleftarrow{D_y} p_2(i,j),$$

where $\overleftarrow{D_x}$ and $\overleftarrow{D_y}$ are the following backward difference operators:

$$\overleftarrow{D_x} u(i,j) = \begin{cases} u(i,j) - u(i,m) & \text{if } 1 \le i \le n,\ j = 1, \\ u(i,j) - u(i,j-1) & \text{if } 1 \le i \le n,\ 1 < j \le m, \end{cases}$$

$$\overleftarrow{D_y} u(i,j) = \begin{cases} u(i,j) - u(n,j) & \text{if } i = 1,\ 1 \le j \le m, \\ u(i,j) - u(i-1,j) & \text{if } 1 < i \le n,\ 1 \le j \le m. \end{cases}$$

Analogously we define the second order discrete differential operators $D_{xx}u$, $D_{yy}u$, $D_{xy}u$, $D_{yx}u$. These are just the corresponding compositions of the first order operators,

$$D_{xx} u(i,j) = \begin{cases} u(i,m) - 2u(i,j) + u(i,j+1) & \text{if } 1 \le i \le n,\ j = 1, \\ u(i,j-1) - 2u(i,j) + u(i,j+1) & \text{if } 1 \le i \le n,\ 1 < j < m, \\ u(i,j-1) - 2u(i,j) + u(i,1) & \text{if } 1 \le i \le n,\ j = m, \end{cases}$$

$$D_{yy} u(i,j) = \begin{cases} u(n,j) - 2u(i,j) + u(i+1,j) & \text{if } i = 1,\ 1 \le j \le m, \\ u(i-1,j) - 2u(i,j) + u(i+1,j) & \text{if } 1 < i < n,\ 1 \le j \le m, \\ u(i-1,j) - 2u(i,j) + u(1,i) & \text{if } i = n,\ 1 \le j \le m, \end{cases}$$

$$D_{xy} u(i,j) = \begin{cases} u(i,j) - u(i+1,j) - u(i,j+1) + u(i+1,j+1) & \text{if } 1 \le i < n,\ 1 \le j < m, \\ u(i,j) - u(1,j) - u(i,j+1) + u(1,j+1) & \text{if } i = n,\ 1 \le j < m, \\ u(i,j) - u(i+1,j) - u(i,1) + u(i+1,1) & \text{if } 1 \le i < n,\ j = m, \\ u(i,j) - u(1,j) - u(i,1) + u(1,1) & \text{if } i = n,\ j = m. \end{cases}$$

One can easily check that $D_{xy} = D_x(D_y) = D_y(D_x) = D_{yx}$. As before, we also need the discrete second order divergence operator $\mathrm{div}^2 : (\mathbb{R}^{n\times m})^4 \to \mathbb{R}^{n\times m}$ with the adjointness property

$$\mathrm{div}^2 q \cdot u = q \cdot \nabla^2 u, \quad \forall u \in \mathbb{R}^{n\times m},\ q \in \left(\mathbb{R}^{n\times m}\right)^4.$$

For a $q = (q_{11}, q_{22}, q_{12}, q_{21}) \in (\mathbb{R}^{n\times m})^4$ we define

$$(\mathrm{div}^2 q)(i,j) = \overleftarrow{D_{xx}} q_{11}(i,j) + \overleftarrow{D_{yy}} q_{22}(i,j) + \overleftarrow{D_{xy}} q_{12}(i,j) + \overleftarrow{D_{yx}} q_{21}(i,j),$$

where

$$\overleftarrow{D_{xx}} = D_{xx}, \quad \overleftarrow{D_{yy}} = D_{yy}, \quad \overleftarrow{D_{xy}} = \overleftarrow{D_{yx}} \quad \text{with}$$

$$\overleftarrow{D_{xy}} u(i,j) = \begin{cases} u(i,j) - u(i,m) - u(n,i) + u(n,m) & \text{if } i = 1,\ j = 1, \\ u(i,j) - u(i,j-1) - u(n,j) + u(n,j-1) & \text{if } i = 1,\ 1 < j \leq m, \\ u(i,j) - u(i-1,j) - u(i,m) + u(i-1,m) & \text{if } 1 < i \leq m\ j = 1, \\ u(i,j) - u(i,j-1) - u(i-1,j) + u(i-1,j-1) & \text{if } 1 < i \leq n,\ 1 < j \leq m. \end{cases}$$

Figure 4 describes the action of all the above discrete differential operators. Having defined all necessary discrete quantities, we now discuss the numerical optimisation of (4).

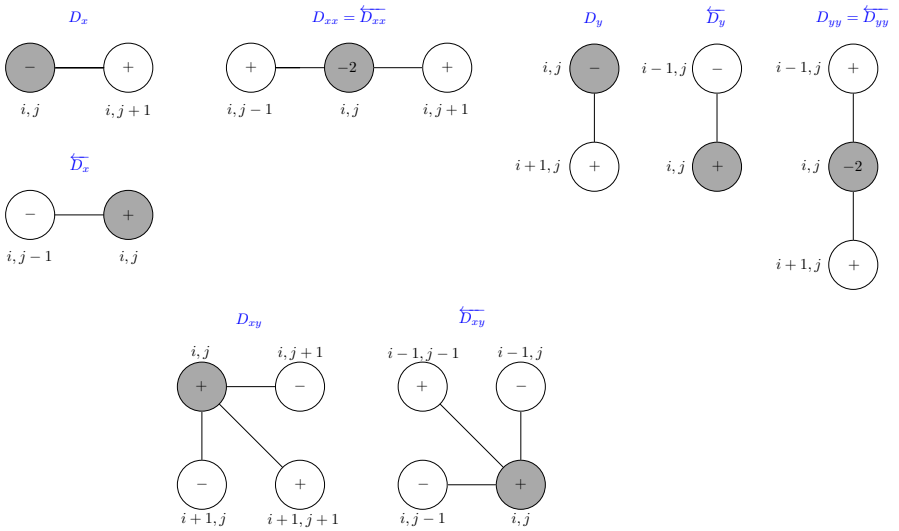



Figure 4: Illustration of the discrete derivative approximations.

## 2.2 The Split Bregman Iteration

In the following, we use what is frequently called split Bregman iteration to solve (4). To do so, we first formulate the unconstrained minimisation problem (4) as a constrained problem and then apply Bregman iteration to solve it [6, 19, 28]. One of the advantages of this transformation is that the problem can be simplified further by splitting it into subproblems which are easier to solve. The corresponding constrained minimisation problem is

$$\min_{\substack{u\in\mathbb{R}^{n\times m}, \tilde{u}\in\mathbb{R}^{n\times m}, \\ v\in(\mathbb{R}^{n\times m})^2, w\in(\mathbb{R}^{n\times m})^4}} \|\mathcal{X}_{\Omega\setminus D}(u - u_0)\|_2^2 + \alpha\|v\|_1 + \beta\|w\|_1 \text{ such that } u = \tilde{u},\ v = \nabla\tilde{u},\ w = \nabla^2\tilde{u}, \qquad (5)$$

where $v = (v_x, v_y)$, $w = (w_{xx}, w_{yy}, w_{xy}, w_{yx})$.[1] Note that the problems (4) and (5) are equivalent. The introduction of the auxiliary variable $\tilde{u}$ is crucial since it allows the use of fast Fourier transform,

[1] Note that this notation denotes coordinates, not derivatives, of $v$ and $w$.

leading to a faster implementation, see Remark 1, page 121. The Bregman iteration that corresponds to (5) is algorithm 1 where $b_1^k = (b_{1,x}^k, b_{1,y}^k) \in (\mathbb{R}^{n\times m})^2$, $b_2^k = (b_{2,xx}^k, b_{2,yy}^k, b_{2,xy}^k, b_{2,yx}^k) \in (\mathbb{R}^{n\times m})^4$ and $\lambda_0, \lambda_1, \lambda_2$ are positive constants.

**Algorithm 1:** Bregman Iteration – 1st and 2nd Order Total Variation Inpainting

Initialisation: $u^1 = u_0$, $\tilde{u}^1 = u_0$, $v^1 = \nabla u_0$, $w^1 = \nabla^2 u_0$, $b_0^1 = \mathbf{1}$, $b_1^1 = \mathbf{1}$, $b_2^1 = \mathbf{1}$

$$\begin{aligned}
[u^{k+1}, \tilde{u}^{k+1}, v^{k+1}, w^{k+1}] &= \underset{u,\tilde{u},v,w}{\operatorname{argmin}} \ \|\mathcal{X}_{\Omega\setminus D}(u-u_0)\|_2^2 + \alpha\|v\|_1 + \beta\|w\|_1 \\
&\quad + \frac{\lambda_0}{2}\|b_0^k + \tilde{u} - u\|_2^2 + \frac{\lambda_1}{2}\|b_1^k + \nabla\tilde{u} - v\|_2^2 \\
&\quad + \frac{\lambda_2}{2}\|b_2^k + \nabla^2\tilde{u} - w\|_2^2, \qquad (6)\\
b_0^{k+1} &= b_0^k + \tilde{u}^{k+1} - u^{k+1}, \qquad (7)\\
b_1^{k+1} &= b_1^k + \nabla\tilde{u}^{k+1} - v^{k+1}, \qquad (8)\\
b_2^{k+1} &= b_2^k + \nabla^2\tilde{u}^{k+1} - w^{k+1}, \qquad (9)
\end{aligned}$$

It is difficult to solve the minimisation problem (6) directly, thus in every iteration we split it into four separate subproblems for $u$, $\tilde{u}$, $v$, $w$, each of which can be solved quickly. This splitting of Bregman iteration forms the split Bregman method that was introduced in Goldstein and Osher [16] for total variation minimisation. See also the works by Getreuer [21, 22, 23] which use the split Bregman method for TV denoising, deblurring and inpainting respectively. In our case the split Bregman formulation reads as detailed in algorithm 2.

**Algorithm 2:** Split Bregman Iteration – 1st and 2nd Order Total Variation Inpainting Grey Scale Images

Initialisation: $u^1 = u_0$, $\tilde{u}^1 = u_0$, $v^1 = \nabla u_0$, $w^1 = \nabla^2 u_0$, $b_0^1 = \mathbf{1}$, $b_1^1 = \mathbf{1}$, $b_2^1 = \mathbf{1}$

$$\begin{aligned}
\textbf{Subproblem 1:}\quad u^{k+1} &= \underset{u\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \|\mathcal{X}_{\Omega\setminus D}(u-u_0)\|_2^2 + \frac{\lambda_0}{2}\|b_0^k + \tilde{u}^k - u\|_2^2, \qquad (10)\\
\textbf{Subproblem 2:}\quad \tilde{u}^{k+1} &= \underset{\tilde{u}\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \frac{\lambda_0}{2}\|b_0^k + \tilde{u} - u^{k+1}\|_2^2 + \frac{\lambda_1}{2}\|b_1^k + \nabla\tilde{u} - v^k\|_2^2 \qquad (11)\\
&\quad + \frac{\lambda_2}{2}\|b_2^k + \nabla^2\tilde{u} - w^k\|_2^2,\\
\textbf{Subproblem 3:}\quad v^{k+1} &= \underset{v\in(\mathbb{R}^{n\times m})^2}{\operatorname{argmin}} \ \alpha\|v\|_1 + \frac{\lambda_1}{2}\|b_1^k + \nabla\tilde{u}^{k+1} - v\|_2^2, \qquad (12)\\
\textbf{Subproblem 4:}\quad w^{k+1} &= \underset{w\in(\mathbb{R}^{n\times m})^4}{\operatorname{argmin}} \ \beta\|w\|_1 + \frac{\lambda_2}{2}\|b_2^k + \nabla^2\tilde{u}^{k+1} - w\|_2^2, \qquad (13)\\
\textbf{Update on } \mathbf{b_0}\textbf{:}\quad b_0^{k+1} &= b_0^k + \tilde{u}^{k+1} - u^{k+1}, \qquad (14)\\
\textbf{Update on } \mathbf{b_1}\textbf{:}\quad b_1^{k+1} &= b_1^k + \nabla\tilde{u}^{k+1} - v^{k+1}, \qquad (15)\\
\textbf{Update on } \mathbf{b_2}\textbf{:}\quad b_2^{k+1} &= b_2^k + \nabla^2\tilde{u}^{k+1} - w^{k+1}. \qquad (16)
\end{aligned}$$

We now describe how we solve each of the four subproblems (10)–(13). The formulae for their

solutions are enclosed in boxes.

### 2.2.1 Subproblem 1

This problem has a closed form solution, so it can be solved very fast. We have

$$\begin{aligned} u^{k+1} &= \underset{u\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \|\mathcal{X}_{\Omega\setminus D}(u-u_0)\|_2^2 + \frac{\lambda_0}{2}\|b_0^k+\tilde{u}^k-u\|_2^2 \\ &= \underset{u\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \sum_{i,j} \mathcal{X}_{\Omega\setminus D}(u(i,j)-u_0(i,j))^2 + \frac{\lambda_0}{2}\left(b_0^k(i,j)+\tilde{u}^k(i,j)-u(i,j)\right)^2 \Rightarrow \\ u^{k+1}(i,j) &= \underset{z\in\mathbb{R}}{\operatorname{argmin}} \ \mathcal{X}_{\Omega\setminus D}(z-u_0(i,j))^2 + \frac{\lambda_0}{2}\left(b_0^k(i,j)+\tilde{u}^k(i,j)-z\right)^2. \end{aligned}$$

It is trivial to calculate the solution of the last minimisation problem. We have

$$\boxed{u^{k+1}(i,j) = \left(\frac{2\mathcal{X}_{\Omega\setminus D}(i,j)}{\lambda_0+2}\right)u_0(i,j) + \left(\frac{\lambda_0+2(1-\mathcal{X}_{\Omega\setminus D}(i,j))}{\lambda_0+2}\right)(b_0^k(i,j)+\tilde{u}^k(i,j)).} \tag{17}$$

### 2.2.2 Subproblem 2

We have

$$\tilde{u}^{k+1} = \underset{\tilde{u}\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \frac{\lambda_0}{2}\|b_0^k+\tilde{u}-u^{k+1}\|_2^2 + \frac{\lambda_1}{2}\|b_1^k+\nabla\tilde{u}-v^k\|_2^2, + \frac{\lambda_2}{2}\|b_2^k+\nabla^2\tilde{u}-w^k\|_2^2.$$

This problem is solved through its optimality condition. In the continuous setting that results in a fourth order linear PDE. In the discrete setting the optimality condition reads as follows:

$$\begin{aligned} &\lambda_0\tilde{u}^{k+1} - \lambda_1\left(\overleftarrow{D_x}(D_x(\tilde{u}^{k+1})) + \overleftarrow{D_y}(D_y(\tilde{u}^{k+1}))\right) \\ &+\lambda_2\left(\overleftarrow{D_{xx}}(D_{xx}(\tilde{u}^{k+1})) + \overleftarrow{D_{yy}}(D_{yy}(\tilde{u}^{k+1})) + 2\overleftarrow{D_{xy}}(D_{xy}(\tilde{u}^{k+1}))\right) \\ &= \\ &\lambda_0(u^{k+1}-b_0^k) + \lambda_1\left(\overleftarrow{D_x}(b_{1,x}^k - v_x^k) + \overleftarrow{D_y}(b_{1,y}^k - v_y^k)\right) \\ &-\lambda_2\left(\overleftarrow{D_{xx}}(b_{2,xx}^k - w_{xx}^k) + \overleftarrow{D_{yy}}(b_{2,yy}^k - w_{yy}^k) + 2\overleftarrow{D_{xy}}(b_{2,xy}^k - w_{xy}^k)\right). \end{aligned} \tag{18}$$

Taking into account that the discrete differential operators act like circular convolutions, we can apply 2-dimensional discrete Fourier transforms in (18) (denoted by $\mathcal{F}(\cdot)$) and get

$$FD\cdot\mathcal{F}(\tilde{u}^{k+1}) = \mathcal{F}(R^k), \tag{19}$$

where

$$FD := \lambda_0 I - \lambda_1\left(\mathcal{F}(\overleftarrow{D_x}(D_x)) + \mathcal{F}(\overleftarrow{D_y}(D_y))\right) + \lambda_2\left(\mathcal{F}(\overleftarrow{D_{xx}}(D_{xx})) + \mathcal{F}(\overleftarrow{D_{yy}}(D_{yy})) + \mathcal{F}(\overleftarrow{D_{xy}}(D_{xy}))\right),$$

and $R^k$ is the righthand side of (18) Here "$\cdot$" means pointwise multiplication of matrices. The term FD needs to be calculated only once at the beginning of the algorithm. Thus, we have

$$\boxed{\tilde{u}^{k+1} = \mathcal{F}^{-1}\left(\mathcal{F}(R^k)/FD\right),} \tag{20}$$

where "/" denotes pointwise division of matrices.

**Remark 1** Here one can notice the utility of the auxiliary variable $\tilde{u}$. Without this variable, the minimisation problem (10) would be absent and the problem (11) would have the form

$$u^{k+1} = \underset{u\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \|\mathcal{X}_{\Omega\setminus D}(u-u_0)\|_2^2 + \frac{\lambda_1}{2}\|b_1^k + \nabla u - v^k\|_2^2 + \frac{\lambda_2}{2}\|b_2^k + \nabla^2 u - w^k\|_2^2. \tag{21}$$

However, we cannot take advantage of the fast Fourier transform in order to solve the optimality condition of (21) as in general $\mathcal{F}(\mathcal{X}_{\Omega\setminus D}u) \neq \mathcal{X}_{\Omega\setminus D}\mathcal{F}(u)$.

### 2.2.3 Subproblem 3

Subproblem 3 has a closed form solution as well. We have

$$\begin{aligned}
v^{k+1} &= \underset{v\in(\mathbb{R}^{n\times m})^2}{\operatorname{argmin}} \ \alpha\|v\|_1 + \frac{\lambda_1}{2}\|b_1^k + \nabla\tilde{u}^{k+1} - v\|_2^2 \\
&= \underset{v\in(\mathbb{R}^{n\times m})^2}{\operatorname{argmin}} \ \sum_{i,j}\alpha\sqrt{v_x(i,j)^2 + v_y(i,j)^2} \\
&\quad + \frac{\lambda_1}{2}\big((b_{1,x}^k(i,j) + D_x\tilde{u}^{k+1}(i,j) - v_x(i,j))^2 \\
&\quad + (b_{1,y}^k(i,j) + D_y\tilde{u}^{k+1}(i,j) - v_y(i,j))^2\big) \Rightarrow \\
v^{k+1}(i,j) &= \underset{(z_1,z_2)\in\mathbb{R}^2}{\operatorname{argmin}} \ \sum_{i,j}\alpha\sqrt{z_1^2 + z_2^2} \\
&\quad + \frac{\lambda_1}{2}\left((b_{1,x}^k(i,j) + D_x\tilde{u}^{k+1}(i,j) - z_1)^2 + (b_{1,y}^k(i,j) + D_y\tilde{u}^{k+1}(i,j) - z_2)^2\right).
\end{aligned}$$

Denoting with $s^k(i,j) = (s_1^k(i,j), s_2^k(i,j)) = \left(b_{1,x}^k(i,j) + D_x\tilde{u}^{k+1}(i,j), b_{1,y}^k(i,j) + D_y\tilde{u}^{k+1}(i,j)\right) \in \mathbb{R}^2$ it is easy to check that

$$\boxed{\begin{aligned}
v_x^{k+1}(i,j) &= \max\left(|s^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right)\tfrac{s_1^k(i,j)}{|s^k(i,j)|}, \\
v_y^{k+1}(i,j) &= \max\left(|s^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right)\tfrac{s_2^k(i,j)}{|s^k(i,j)|},
\end{aligned}} \tag{22}$$

with the convention that $0/0 = 0$. The above operation on (22) is known as *shrinkage*.

### 2.2.4 Subproblem 4

We have

$$w^{k+1} = \underset{w\in(\mathbb{R}^{n\times m})^4}{\operatorname{argmin}} \ \beta\|w\|_1 + \frac{\lambda_2}{2}\|b_2^k + \nabla^2\tilde{u}^{k+1} - w\|_2^2.$$

Working in the same way as in subproblem 3 and setting

$$\begin{aligned}
t^k(i,j) &= (t_1^k(i,j), t_2^k(i,j), t_3^k(i,j), t_3^k(i,j)) \\
&= \big(b_{2,xx}^k(i,j) + D_{xx}\tilde{u}^{k+1}(i,j), b_{2,yy}^k(i,j) + D_{yy}\tilde{u}^{k+1}(i,j), \\
&\quad b_{2,xy}^k(i,j) + D_{xy}\tilde{u}^{k+1}(i,j), b_{2,xy}^k(i,j) + D_{xy}\tilde{u}^{k+1}(i,j)\big) \in \mathbb{R}^4,
\end{aligned}$$

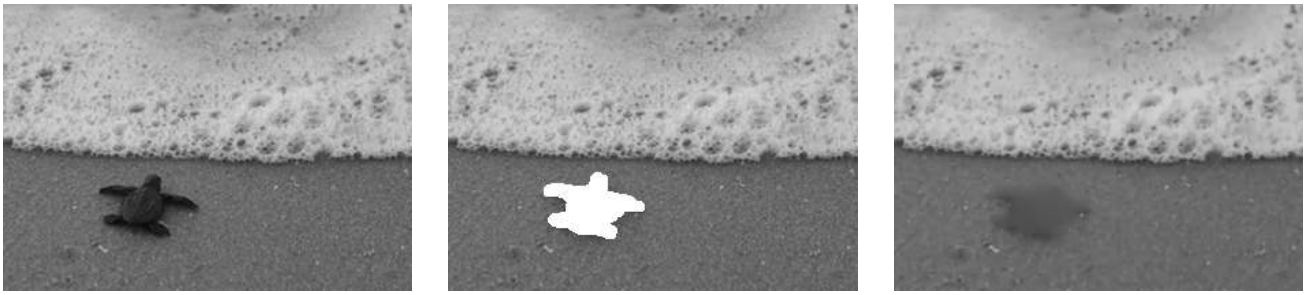
we obtain

$$\boxed{\begin{aligned}
w_{xx}^{k+1}(i,j) &= \max\left(|t^k(i,j)| - \tfrac{\beta}{\lambda_2}, 0\right)\tfrac{t_1^k(i,j)}{|t^k(i,j)|}, \\
w_{yy}^{k+1}(i,j) &= \max\left(|t^k(i,j)| - \tfrac{\beta}{\lambda_2}, 0\right)\tfrac{t_2^k(i,j)}{|t^k(i,j)|}, \\
w_{xy}^{k+1}(i,j) &= \max\left(|t^k(i,j)| - \tfrac{\beta}{\lambda_2}, 0\right)\tfrac{t_3^k(i,j)}{|t^k(i,j)|}.
\end{aligned}} \tag{23}$$

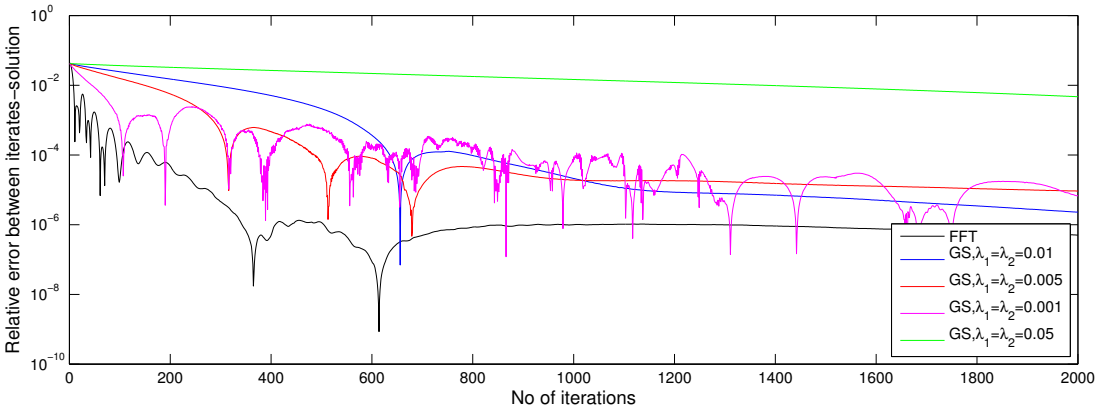
**Remark 2** Let us note here that all four subproblems (10)–(13) are solved exactly. This is in contrast with several previous works [16, 21, 22, 23] where the subproblem of the type (11) is solved approximately with one iteration of Gauss–Seidel to the linear system that results from the optimality condition. There, the optimality condition leads to a second order linear PDE instead of fourth order. If one chooses to use Gauss–Seidel for our method then there is no need to use the auxiliary variable $\tilde{u}$ and the corresponding linear problem reads

$$u^{k+1} = \underset{u\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \ \|\mathcal{X}_{\Omega\setminus D}(u-u_0)\|_2^2 + \frac{\lambda_1}{2}\|b_1^k + \nabla u - v^k\|_2^2 + \frac{\lambda_2}{2}\|b_2^k + \nabla^2 u - w^k\|_2^2. \tag{24}$$

Even though convergence is achieved using both methods, the FFT approach leads to a faster and a more robust convergence, see figure 5 for a quantitative comparison.



(a) Original image. (b) Inpainting domain (c) Inpainted image with $\alpha = \beta = 0.001$.




(d) Plot of the relative error $|\|u_k - \text{sol}\|_2|/\|\text{sol}\|_2$ versus the number of iterations. Here *sol* denotes a ground solution which is taken to be the solution $u_k$ for a large enough $k$ (3000).

Figure 5: We compare the efficiency of the fast Fourier transform for the solution of (18) and one iteration of Gauss–Seidel for the solution of (24). For the FFT the choice of $\lambda$'s is done according to (44) while for the Gauss–Seidel the selection of $\lambda$'s is shown in figure (d). We observe that solving the linear subproblem with FFT leads to a faster convergence as figure (d) suggests. Using more that one iterations of Gauss–Seidel do not improve the situation significantly and also increase the computational time, see also the corresponding discussion in Goldstein and Osher [16].

## 2.3 The Coloured Image Case

So far we have considered greyscale images only, i.e., $u \in \mathbb{R}^{n\times m}$. However it is easy to derive a version of the algorithm (10)–(16) for coloured images, that is to say when $\mathbf{u} \in (\mathbb{R}^3)^{n\times m}$, where for

simplicity we only consider RGB images. Starting from the continuous setting, if $\mathbf{u} = (u_1, u_2, u_3) = (u, u, u) \in \mathrm{BV}(\Omega; \mathbb{R}^3)$, then the total variation of $u$ is defined as

$$\mathrm{TV}(\mathbf{u}) = \sup \left\{ \sum_{i=1}^{3} \int_\Omega u_i \mathrm{div} \phi_i : \quad \phi \in \left[C_c^1(\Omega, \mathbb{R}^2)\right]^3, \ \|\phi\|_\infty \leq 1 \right\}. \tag{25}$$

As in the scalar case, if $\mathbf{u} \in W^{1,1}(\Omega, \mathbb{R}^3)$, then

$$\mathrm{TV}(\mathbf{u}) = \int_\Omega |\nabla \mathbf{u}| \, dx. \tag{26}$$

Analogously we define $\mathrm{TV}^2(\mathbf{u})$. In the discrete setting the first and second order total variation have the form

$$\begin{aligned} \mathrm{TV}(\mathbf{u}) &= \|(\nabla u, \nabla u, \nabla u)\|_1, & (27) \\ \mathrm{TV}^2(\mathbf{u}) &= \|(\nabla^2 u, \nabla^2 u, \nabla^2 u)\|_1. & (28) \end{aligned}$$

The minimisation problem corresponding to (4) is

$$\min_{\mathbf{u} \in (\mathbb{R}^3)^{n \times m}} \|\mathcal{X}_{\Omega \setminus D}(\mathbf{u} - \mathbf{u}_0)\|_2^2 + \alpha \|\nabla \mathbf{u}\|_1 + \beta \|\nabla^2 \mathbf{u}\|_1. \tag{29}$$

It is not hard to check that the only difference to the minimisation algorithm of the greyscale case is in the subproblems (12) and (13). This is the only point where the colours are mixed. Subproblems (10), (11) and the updates (14)–(16) are done separately for each channel. This favours parallelisation of the algorithm 3.

---

**Algorithm 3:** Split Bregman Iteration – 1st and 2nd order Total Variation Inpainting Coloured Images

---

Initialisation: $\mathbf{u}^1 = \mathbf{u}_0$, $\tilde{\mathbf{u}}^1 = \mathbf{u}_0$, $\mathbf{v}^1 = \nabla \mathbf{u}_0$, $\mathbf{w}^1 = \nabla^2 \mathbf{u}_0$, $\mathbf{b}_0^1 = \mathbf{1}$, $\mathbf{b}_1^1 = \mathbf{1}$, $\mathbf{b}_2^1 = \mathbf{1}$

**Subproblem 1:** Solve (10) separately for $u^{k+1}, u^{k+1}, u^{k+1}$, (30)

**Subproblem 2:** Solve (11) separately for $\tilde{u}^{k+1}, \tilde{u}^{k+1}, \tilde{u}^{k+1}$, (31)

**Subproblem 3:**
$$\mathbf{v}^{k+1} = \operatorname*{argmin}_{\mathbf{v} \in \left((\mathbb{R}^{n \times m})^2\right)^3} \alpha \|\mathbf{v}\|_1 + \frac{\lambda_1}{2} \|\mathbf{b}_1^k + \nabla \tilde{\mathbf{u}}^{k+1} - \mathbf{v}\|_2^2, \tag{32}$$

**Subproblem 4:**
$$\mathbf{w}^{k+1} = \operatorname*{argmin}_{\mathbf{w} \in \left((\mathbb{R}^{n \times m})^4\right)^3} \beta \|\mathbf{w}\|_1 + \frac{\lambda_2}{2} \|\mathbf{b}_2^k + \nabla^2 \tilde{\mathbf{u}}^{k+1} - \mathbf{w}\|_2^2, \tag{33}$$

**Update on $\mathbf{b_0}$:** Do update (14) separately for $b_0^{k+1}, b_0^{k+1}, b_0^{k+1}$, (34)

**Update on $\mathbf{b_1}$:** Do update (15) separately for $b_1^{k+1}, b_1^{k+1}, b_1^{k+1}$, (35)

**Update on $\mathbf{b_2}$:** Do update (16) separately for $b_2^{k+1}, b_2^{k+1}, b_2^{k+1}$. (36)

---

As far as the solution of (32) is concerned, similarly as before we set

$$\begin{aligned} \mathbf{s}^k(i,j) &= (s_1^k(i,j), s_2^k(i,j), s_1^k(i,j), s_2^k(i,j), s_1^k(i,j), s_2^k(i,j)) \\ &= \Big(b_{1,x}^k(i,j) + D_x \tilde{u}^{k+1}(i,j), b_{1,y}^k(i,j) + D_y \tilde{u}^{k+1}(i,j), \\ &\quad b_{1,x}^k(i,j) + D_x \tilde{u}^{k+1}(i,j), b_{1,y}^k(i,j) + D_y \tilde{u}^{k+1}(i,j), \\ &\quad b_{1,x}^k(i,j) + D_x \tilde{u}^{k+1}(i,j), b_{1,y}^k(i,j) + D_y \tilde{u}^{k+1}(i,j)\Big), \end{aligned}$$

and

$$v_x^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_1^k(i,j)}{|\mathbf{s}^k(i,j)|}, \tag{37}$$

$$v_y^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_2^k(i,j)}{|\mathbf{s}^k(i,j)|}, \tag{38}$$

$$v_x^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_1^k(i,j)}{|\mathbf{s}^k(i,j)|}, \tag{39}$$

$$v_y^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_2^k(i,j)}{|\mathbf{s}^k(i,j)|}, \tag{40}$$

$$v_x^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_1^k(i,j)}{|\mathbf{s}^k(i,j)|}, \tag{41}$$

$$v_y^{k+1}(i,j) = \max\left(|\mathbf{s}^k(i,j)| - \tfrac{\alpha}{\lambda_1}, 0\right) \tfrac{s_2^k(i,j)}{|\mathbf{s}^k(i,j)|}. \tag{42}$$

Analogously, we compute the solution of (33).

## 2.4 Stopping Criteria and the Selection of $\lambda$'s

We have found empirically that a stopping criterion depending on the relative residue of $u$ is a good one for the termination of the algorithm. Thus the iterations are terminated when

$$\max\left\{\frac{|u_k - u_{k-1}|}{|u_{k-1}|}, \frac{|u_k - u_{k-1}|}{|u_{k-1}|}, \frac{|u_k - u_{k-1}|}{|u_{k-1}|}\right\} \leq 8 \cdot 10^{-5}. \tag{43}$$

The parameters $\lambda_0$, $\lambda_1$ and $\lambda_2$ have to be chosen carefully. This selection is done automatically in the online demo, so the user does not have to worry about them. All the combinations lead to convergence but the number of iterations that are needed for convergence differ dramatically for different combinations. Empirically we have found that $\lambda_1$ and $\lambda_2$ have to be one or two orders of magnitude larger than $\alpha$ and $\beta$ respectively. The value of $\lambda_0$ should not be much larger than $\alpha$ and $\beta$ as that will lead to a slow convergence. Here we choose the following values of these parameters, a combination that has shown to ensure fast convergence:

$$\lambda_1 = 10\alpha, \quad \lambda_2 = 10\beta, \quad \lambda_0 = \max\{\alpha, \beta\}. \tag{44}$$

Table 1 and figure 6 justify this empirical choice. In figure 6, we perform pure $\mathrm{TV}^2$ inpainting to the image of the example in figure 5 for different choices of $\lambda_0$ and $\lambda_2$ and fixed $\beta = 0.001$. In each case we plot the relative error between the iterates and the solution. The solution is taken to be a large iterate ($n = 4000$) that is obtained with the choice (44). What we generally observe can be seen in table 1. The choice (44) leads to a fast convergence with small oscillatory behaviour in the iterates, see black line in figure 6.

| Choice of $\lambda$'s | Convergence to solution | Qualitative behaviour |
|---|---|---|
| $\lambda_0 = \beta$, $\lambda_2 = 10\beta$ | Fast | Small oscillations |
| $\lambda_0 = \beta$, $\lambda_2 \gg 10\beta$ | Fairly fast | Smooth transition |
| $\lambda_0 = \beta$, $\lambda_2 \ll 10\beta$ | Very slow | Wild oscillations |
| $\lambda_0 \gg \beta$, $\lambda_2 = 10\beta$ | Very slow | Smooth transition |
| $\lambda_0 \ll \beta$, $\lambda_2 = 10\beta$ | Slow | Smooth transition |

Table 1: Different behaviours of the iterates for different choices of the $\lambda$ parameters.

The effect of different choices of $\lambda$'s is discussed further in Appendix A. As it is also discussed there, the stopping criterion (43) works well only in combination with the choices (44). Some choices of $\lambda$'s lead to such a slow convergence that even though criterion (43) might be satisfied, the gap in the inpainting domain is not filled in yet.

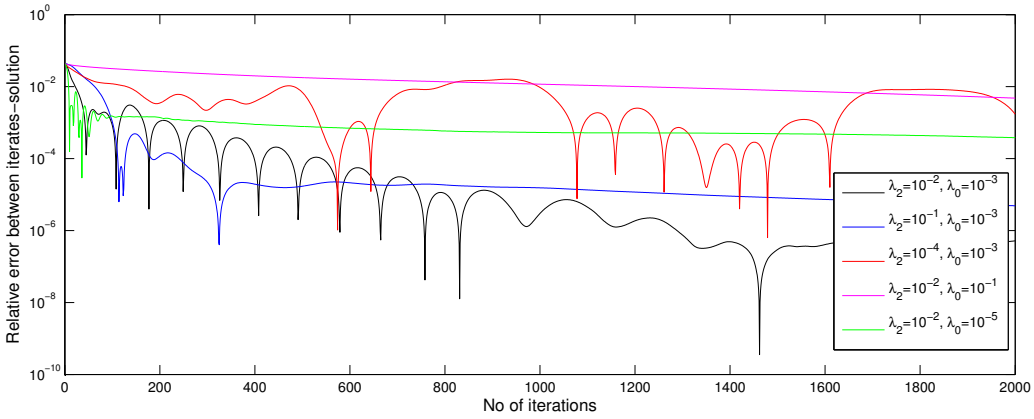



Figure 6: Plot of the relative error of the iterates and the solution (pure TV$^2$ inpainting of the image in figure 5) for different choices of the parameters $\lambda_0$ and $\lambda_2$. The value of $\beta$ is 0.001 and the solution is taken to be the 4000th iterate obtained with $\lambda_0 = 0.001$, $\lambda_2 = 0.01$ – black line. We observe that other choices of $\lambda$'s generally lead to a slower convergence.

# 3 Inpainting Examples

## 3.1 General Examples

In this section we give some inpainting examples. Mainly we want to emphasise the differences in practice between pure first order ($\beta = 0$) and pure second order ($\alpha = 0$) total variation inpainting.

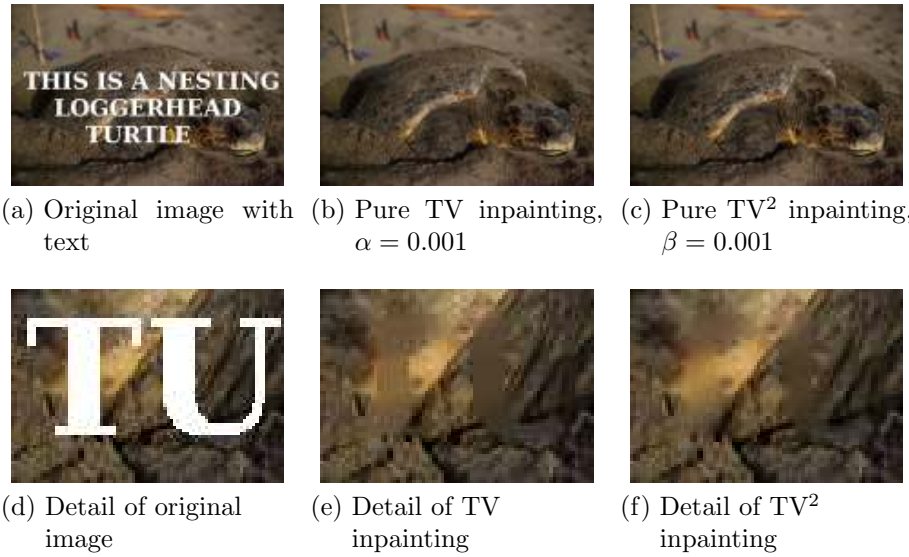
In figure 7 we see a first example where we try to remove a large font text from a natural image. The TV$^2$ result seems more pleasant to the human eye (subjective) as TV inpainting produces piecewise constant parts in the inpainting domain. This is confirmed quantitatively by the PSNR values inside the inpainting domain, as these are higher for the TV$^2$ inpainting.

In figure 8 we see a second example with coloured stripes. Here we include also a harmonic inpainting example. As expected, in the case of TV inpainting the image is piecewise constant inside the inpainting domain, see figure 8(c). This produces a desirable result in stripes whose width is larger than the size of the gap, connecting the gap and preserving the edges there, while TV$^2$ inpainting, figure 8(d), adds additional blur at the edge of the stripe that belongs to the inpainting domain. TV inpainting fails to connect the thin stripes while TV$^2$ inpainting more reliably propagates the right colour, see figures 8(e) and 8(f). Finally, notice the difference between harmonic and TV$^2$ inpainting in terms of connectedness, see for example the yellow stripe of figures 8(g) and 8(h).

We also want to examine how the inpainting result behaves while we are moving from pure TV to pure TV$^2$ inpainting in a continuous way. We can see this in figure 9, where, keeping the geometry of the inpainting domain fixed, we see how connectedness is achieved while we are making the transition from pure TV to pure TV$^2$. The parameters $\alpha$ and $\beta$ vary as follows:

$$\begin{aligned} \alpha &: 0.01 \to 0.008 \to 0.006 \to 0.005 \to 0.004 \to 0.002 \to 0, \\ \beta &: 0 \to 0.002 \to 0.004 \to 0.005 \to 0.006 \to 0.008 \to 0.01. \end{aligned}$$

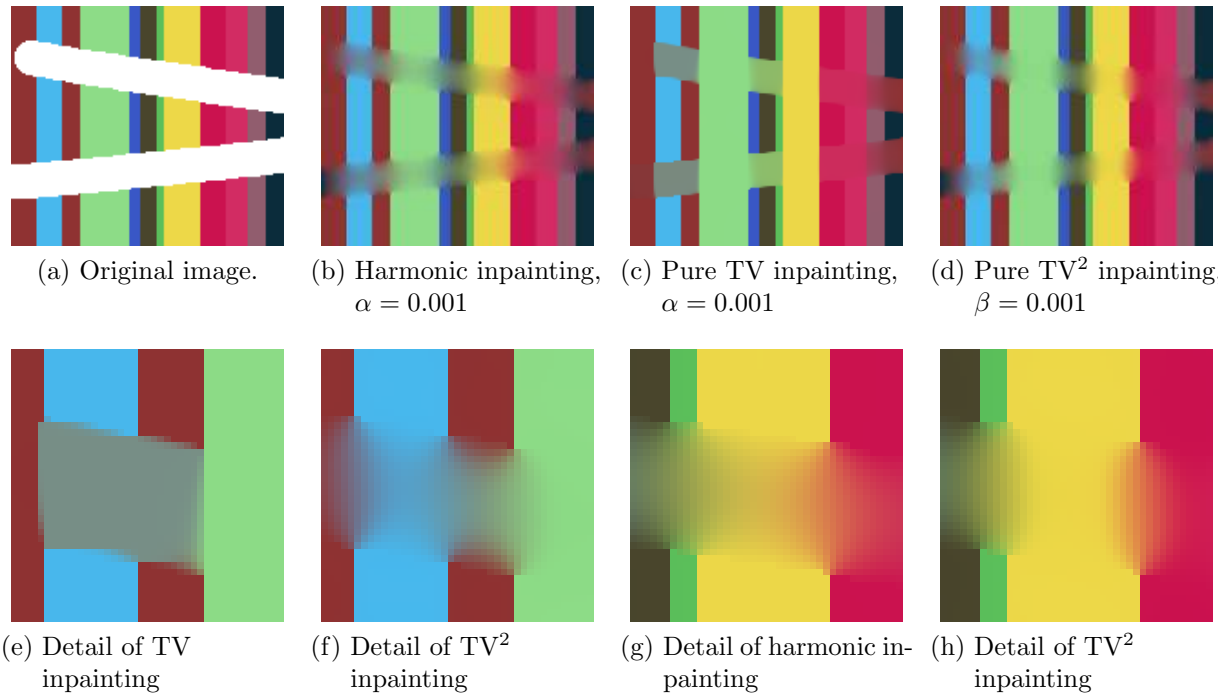
As we see in figure 9, the large gap in the stripe is connected only for large values of $\beta$ and small values of $\alpha$, i.e., for large values of the ratio $\beta/\alpha$. In the case where both weights $\alpha$ and $\beta$ are equal we observe no connection.




(a) Original image with text (b) Pure TV inpainting, $\alpha = 0.001$ (c) Pure TV$^2$ inpainting, $\beta = 0.001$

(d) Detail of original image (e) Detail of TV inpainting (f) Detail of TV$^2$ inpainting

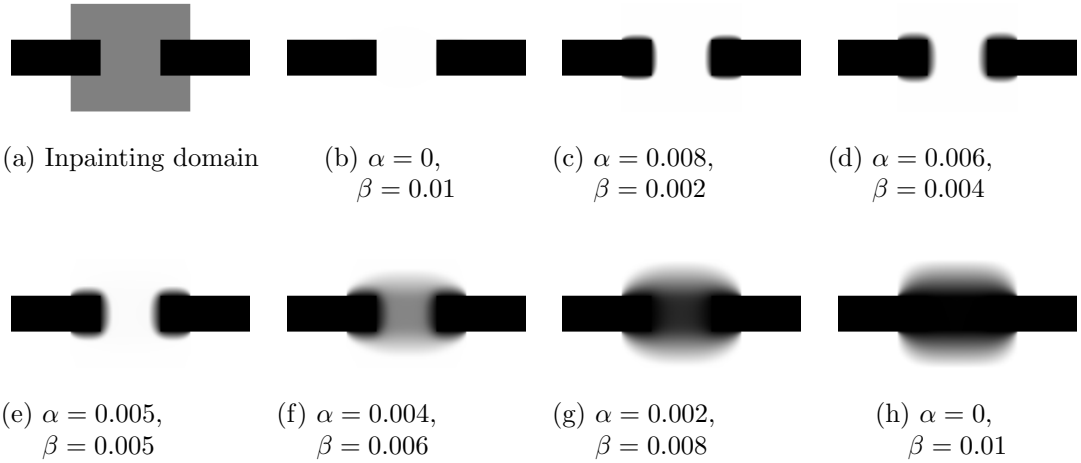
Figure 7: Removing large font text from an image. Even though this is subjective, we believe that the pure TV$^2$ result looks more realistic since the piecewise constant TV reconstruction is not desirable for natural images, for example in inpainting of the letter "T". The PSNR inside the inpainting domain for the three channels is $(34.19, 34.97, 36.99)$ for the TV inpainting and $(34.74, 35.34, 37.20)$ for the TV$^2$ inpainting.



(a) Original image. (b) Harmonic inpainting, $\alpha = 0.001$ (c) Pure TV inpainting, $\alpha = 0.001$ (d) Pure TV$^2$ inpainting, $\beta = 0.001$

(e) Detail of TV inpainting (f) Detail of TV$^2$ inpainting (g) Detail of harmonic inpainting (h) Detail of TV$^2$ inpainting

Figure 8: Coloured stripes inpainting. Observe the difference between TV and TV$^2$ inpainting in the light blue stripe at the left, figures (e) and (f), where in the TV$^2$ case the blue colour is propagated inside the inpainting domain.



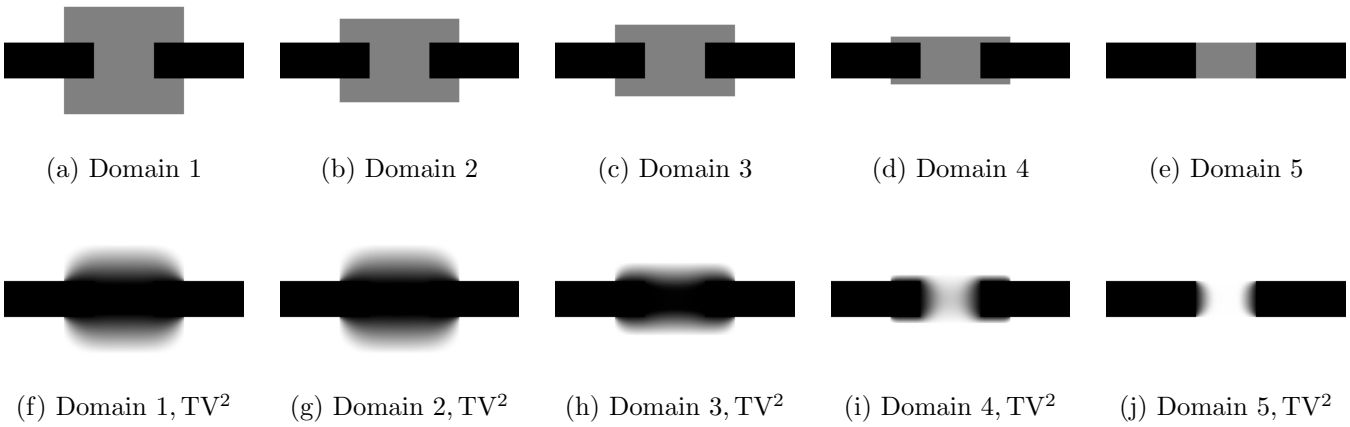
(a) Inpainting domain (b) $\alpha = 0$, $\beta = 0.01$ (c) $\alpha = 0.008$, $\beta = 0.002$ (d) $\alpha = 0.006$, $\beta = 0.004$

(e) $\alpha = 0.005$, $\beta = 0.005$ (f) $\alpha = 0.004$, $\beta = 0.006$ (g) $\alpha = 0.002$, $\beta = 0.008$ (h) $\alpha = 0$, $\beta = 0.01$

Figure 9: Inpainting of a stripe with a large gap. Transition from pure TV to pure $TV^2$. Connectedness is achieved only for large ratio $\beta/\alpha$, see figures (g) and (h). No connectedness is obtained when the weights $\alpha$ and $\beta$ are equal, see figure (e).

## 3.2 The Influence of the Inpainting Domain on Large Gap Connection

In figures 3(l) and 9(h) we saw that the pure $TV^2$ inpainting has the ability to connect large gaps along the inpainting domain. However, as numerical experiments have shown, the quality of the connection depends on the geometry of the inpainting domain $D$, see figures 10 and 11. In particular, for the broken stripe example of figures 10 and 11, the domain must extend above, underneath, left and right of the gap between the two parts of the stripe. For example in figure 10(e), where the inpainting domain is just the area between the two parts, we do not have connection, see figure 10(j), nor we have when we extend the domain only in the vertical direction, figures 11(e) and (j). We have not derived any sufficient conditions on the geometry of $D$ for achieving connectedness and this is a matter of future research.

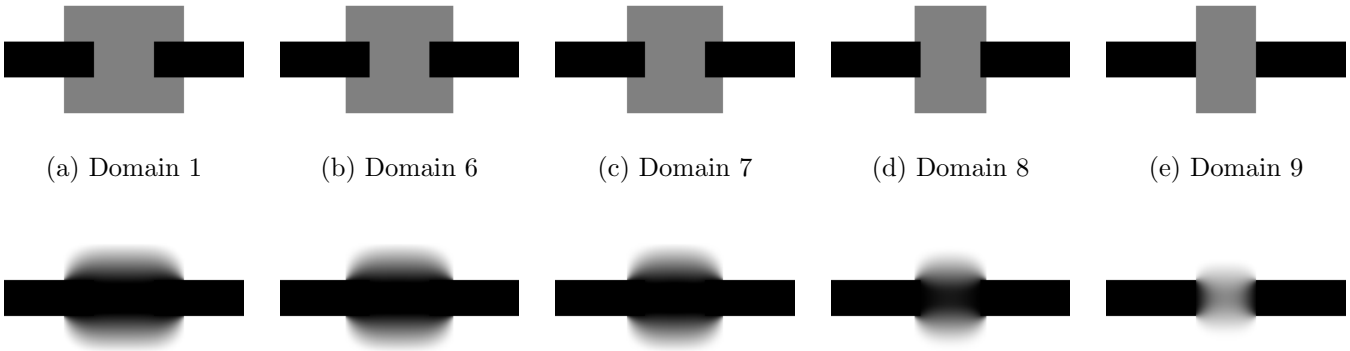


(a) Domain 1 (b) Domain 2 (c) Domain 3 (d) Domain 4 (e) Domain 5

(f) Domain 1, $TV^2$ (g) Domain 2, $TV^2$ (h) Domain 3, $TV^2$ (i) Domain 4, $TV^2$ (j) Domain 5, $TV^2$

Figure 10: Different pure $TV^2$ inpainting results for different inpainting domains of decreasing height. In all computations we set $\beta = 0.001$.

However, let us discuss the following heuristic considerations for the harmonic and biharmonic inpainting results in figures 3 and 12. Both of these inpainting methods are quadratic and hence



(a) Domain 1 (b) Domain 6 (c) Domain 7 (d) Domain 8 (e) Domain 9

(f) Domain 1, $\mathrm{TV}^2$ (g) Domain 6, $\mathrm{TV}^2$ (h) Domain 7, $\mathrm{TV}^2$ (i) Domain 8, $\mathrm{TV}^2$ (j) Domain 9, $\mathrm{TV}^2$

Figure 11: Different pure $\mathrm{TV}^2$ inpainting results for different inpainting domains of decreasing width. In all computations we set $\beta = 0.001$.

possess linear Euler–Lagrange equations, which makes the qualitative study of minimisers easier. In what follows we assume that both $u_0$ and the boundary of the inpainting domain $\partial D$ are sufficiently smooth. For harmonic inpainting the corresponding Euler–Lagrange equation reads

$$-\alpha\Delta u + 2\mathcal{X}_{\Omega\setminus D}(u - u_0) = 0 \quad \text{in } \Omega.$$

Taking the limit $\alpha \to 0$ leads to the following boundary value problem:

$$\begin{aligned} \Delta u &= 0 \quad \text{in } D, \\ u &= u_0 \quad \text{on } \partial D. \end{aligned} \tag{45}$$

Now we can draw the connection to the heat equation. Let $u(x)$ be the temperature in a point $x \in \mathbb{R}^2$. Interpreting the black stripe in figure 3 to be a heating rod that heats up the grey container defined by the inpainting domain $D$, the inpainting results for small $\alpha$ can be explained as the equilibrium heat distribution inside the grey container, i.e., the $u$ that solves (45). While in figure 3(b) the heating rods are only touching the container at the outside of its walls—resulting into a rather modest and almost one-dimensional diffusion of heat in the container—in figure 3(c) the hot rods are stuck into the grey container that should be heated up, resulting into a blob-like diffusion of heat in both horizontal and vertical direction. In this case, the inpainting results are almost the same and no connectedness can be achieved in either case.

Turning our attention to biharmonic inpainting

$$\min_u \int_{\Omega\setminus D} (u - u_0)^2 \, dx + \int_\Omega (\Delta u)^2 \, dx, \tag{46}$$

the situation is different. In this case, the corresponding Euler–Lagrange equation reads

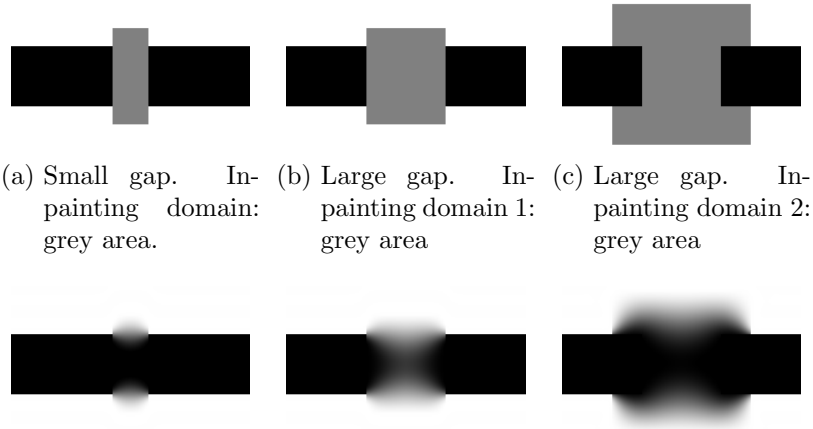
$$\alpha\Delta^2 u + 2\mathcal{X}_{\Omega\setminus D}(u - u_0) = 0 \quad \text{in } \Omega.$$

Applying the same argumentation as for harmonic inpainting above we get the following limiting behaviour

$$\begin{aligned} &\Delta^2 u = 0 && \text{in } D, \\ &u = u_0 && \text{on } \partial D, \\ &+ \text{ appropriate 2nd boundary condition} && \text{on } \partial D. \end{aligned} \tag{47}$$

The biharmonic equation (47) models the bending of an elastic flat plate enforced by fixing it to given values on the boundary $\partial D$. Here, $u(x)$ denotes the displacement of the plate from its initial flat state. Taking the plate model (47) as a good approximation of the biharmonic inpainting model for small $\alpha$, we can now attempt to interpret the inpainting results in figure 12(e) and (f). In the first setup in figure 12(b) the plate is fixed to 0, i.e., the grey value black, on two parts of the vertical boundaries only, and fixed to 255 (white) on the rest of the boundary. The resulting "plate" has a saddle point shape, failing to connect the two ends of the black stripe. On the other hand, in figure 12(c) the plate is fixed to 0 on parts of the vertical and horizontal boundary, where the latter presses the saddle point down to a minimum point at 0 such that the ends of the stripe are (although not through a perfect line) connected this time. Indeed, in contrast to harmonic inpainting, the biharmonic inpainting result seems to depend heavily on the geometry of the inpainting domain $D$.

Going back to the numerical examples for $TV^2$ inpainting in figures 10 and 11 a similar behaviour to the one of biharmonic inpainting can be observed. As before, we see that in order to achieve perfect connectedness it seems crucial that the inpainting domain is larger than the gap between the two stripes. In contrast to biharmonic inpainting though, the nonlinearity within the fourth-order diffusion of the corresponding Euler–Lagrange equation introduces less blur and results into a straighter connection of the line.



(a) Small gap. Inpainting domain: grey area.

(b) Large gap. Inpainting domain 1: grey area

(c) Large gap. Inpainting domain 2: grey area

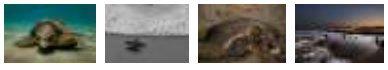
(d) Result of biharmonic inpainting with $\alpha = 0.05$.

(e) Result of biharmonic inpainting with $\alpha = 0.05$.

(f) Result of biharmonic inpainting with $\alpha = 0.05$.

Figure 12: Examples of biharmonic inpainting. The inpainting domain influences the connection of the stripe in a similar way to pure $TV^2$ inpainting (figure 11).

## Image Credits



Konstantinos Papafitsoros, CC-BY

## Acknowledgements

All three authors acknowledge the financial support provided by the Cambridge Centre for Analysis (CCA) funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/H023348/1. The first and third author also acknowledge the financial support of the Royal Society International Exchanges Award IE110314 for the project "High-order Compressed Sensing

for Medical Imaging", the EPSRC / Isaac Newton Trust Small Grant "Non-smooth geometric reconstruction for high resolution MRI imaging of fluid transport in bed reactors" and the EPSRC first grant Nr. EP/J009539/1 "Sparse & Higher-order Image Restoration". Further, this publication is based on work supported by Award No. KUK-I1-007-43, made by King Abdullah University of Science and Technology (KAUST).

# A Effect of Different Combinations of $\lambda$'s on the Speed and the Behaviour of Split Bregman Iteration

As we saw in section 2.4, we follow the rules (44) for the selection of $\lambda_0$, $\lambda_1$ and $\lambda_2$. It was mentioned that this choice was empirical. Different combinations of $\lambda$'s not only result in different speeds of the algorithm but the qualitative behaviour of the solutions is affected as well. For simplicity we study here the case of pure $\mathrm{TV}^2$ inpainting for a fixed parameter $\beta = 0.001$, applied on the images that are shown on figure 13. For each example we provide two intermediate iterates and the final solution.

(a) Original image (b) Destroyed image

Figure 13: Original and damaged image that are used for the comparison of convergence speed and behaviour of split Bregman iterates for different combinations of $\lambda_0$ and $\lambda_2$. For simplicity we perform pure $\mathrm{TV}^2$ inpainting with fixed $\beta = 0.001$.

In the first example, figure 14, we use the combination of $\lambda$'s specified by the rules (44). Convergence is achieved rather quickly, in 482 iterations. We also provide plots of the evolutions of the norms $\|u - \tilde{u}\|_2$, $\|w_{xx} - D_{xx}\tilde{u}\|_2$, $\|w_{yy} - D_{yy}\tilde{u}\|_2$ and $\|w_{xy} - D_{xy}\tilde{u}\|_2$. In order for the split Bregman algorithm to converge, the constraints in (5) should be satisfied, thus these norms should decrease to zero and this is the case for this example. With this combination of $\lambda$'s the iterates exhibit a small oscillatory behaviour.

In the second example, figure 15, we keep the value of $\lambda_0$ fixed and equal to $\beta$ according to (44) and we increase the value of $\lambda_2$ by an order of magnitude. As in the previous case, the constraints in (5) are achieved quickly and the convergence is achieved in 300 iterations. Even though this is faster than in the previous example, increasing the value of $\lambda_2$ tends to slow down the convergence and the iterates are smoother.

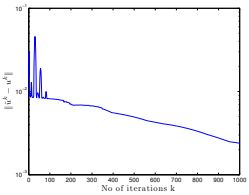
In the third example, figure 16, we keep again the value of $\lambda_0$ fixed but this time we decrease the value of $\lambda_2$ two orders of magnitude, down to $10^{-5}$. This causes a high oscillatory behaviour among the iterates and convergence is not achieved before 13790 iterations. The stopping criterion (43) is satisfied in iteration 11150 but the inpainting domain is not filled in yet. Thus, it is important to use stopping criterion (43) in combination with the choice (44). The reason for the slow convergence is that the constraint $w = D^2u$ is not satisfied quickly, see figures 16(e), (f), (g).

In the fourth example, figure 17, setting $\lambda_2 = 10\beta$ according to 44, we increase the value of $\lambda_0$ by two orders of magnitude up to $10^{-1}$. The main observation here is that even though no oscillations

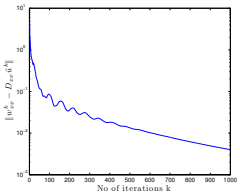
(a) 10 iterations

(b) 100 iterations

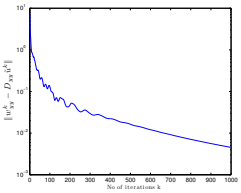
(c) 482 iterations



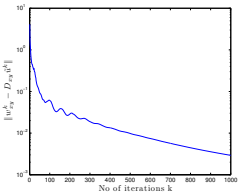
(d) $\|\tilde{u} - u\|_2$



(e) $\|w_{xx} - D_{xx}\tilde{u}\|_2$



(f) $\|w_{yy} - D_{yy}\tilde{u}\|_2$



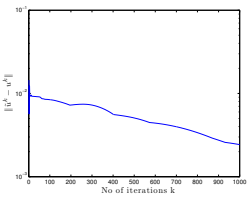
(g) $\|w_{xy} - D_{xy}\tilde{u}\|_2$

Figure 14: $\beta = 0.001$, $\lambda_0 = 0.001$, $\lambda_2 = 0.01$ – choice (44). The convergence is achieved rather quickly in 482 iterations (first time that (43) is achieved). There is a small oscillatory behaviour among the iterates. Also, the constraints in (5) are satisfied since the corresponding norms converge to zero, as figures (d), (e), (f) and (e) indicate.

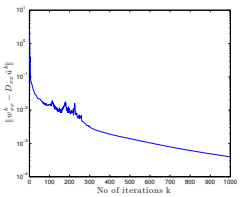
(a) 10 iterations

(b) 100 iterations

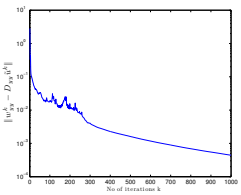
(c) 300 iterations



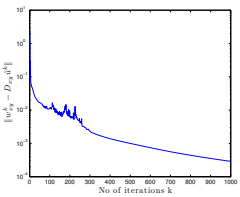
(d) $\|\tilde{u} - u\|_2$



(e) $\|w_{xx} - D_{xx}\tilde{u}\|_2$



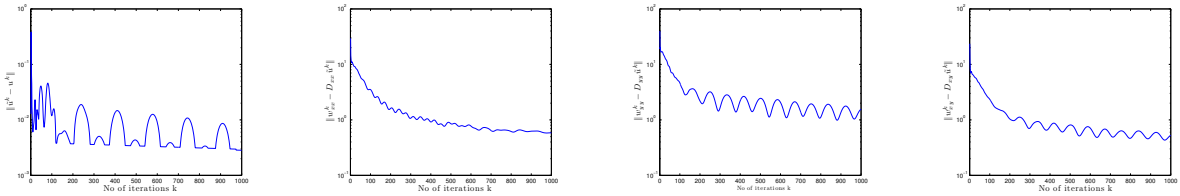
(f) $\|w_{yy} - D_{yy}\tilde{u}\|_2$



(g) $\|w_{xy} - D_{xy}\tilde{u}\|_2$

Figure 15: $\beta = 0.001$, $\lambda_0 = 0.001$, $\lambda_2 = 0.1$. The convergence is achieved very quickly in 300 iterations (first time that (43) is achieved). The convergence is smooth, i.e., no oscillatory behaviour is observed among the iterates. The constraints in (5) are also achieved quickly in this case.

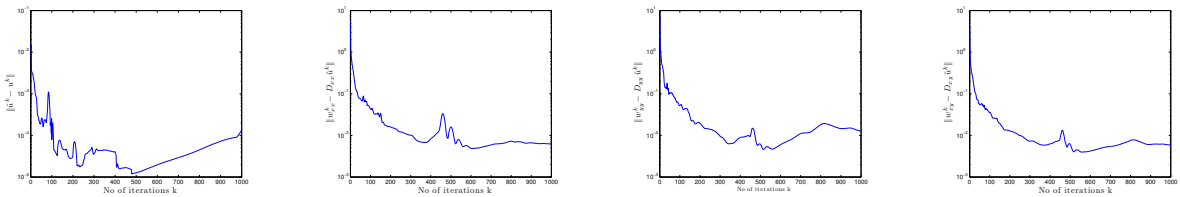
(a) 100 iterations (b) 1000 iterations (c) 13790 iterations



(d) $\|\tilde{u} - u\|_2$ (e) $\|w_{xx} - D_{xx}\tilde{u}\|_2$ (f) $\|w_{yy} - D_{yy}\tilde{u}\|_2$ (g) $\|w_{xy} - D_{xy}\tilde{u}\|_2$

Figure 16: $\beta = 0.001$, $\lambda_0 = 0.001$, $\lambda_2 = 0.00001$. The convergence is very slow, 13790 iterations. In fact (43) is satisfied in iteration 11150 but the inpainting domain is not filled in yet. Wild oscillations are observed throughout the iteration. This is observed generally when $\lambda_2 \ll \beta$. Notice that the constraint $w = D^2u$ is not satisfied quickly, as figures (e), (f) and (g) suggest.

(a) 100 iterations (b) 1000 iterations (c) 6680 iterations

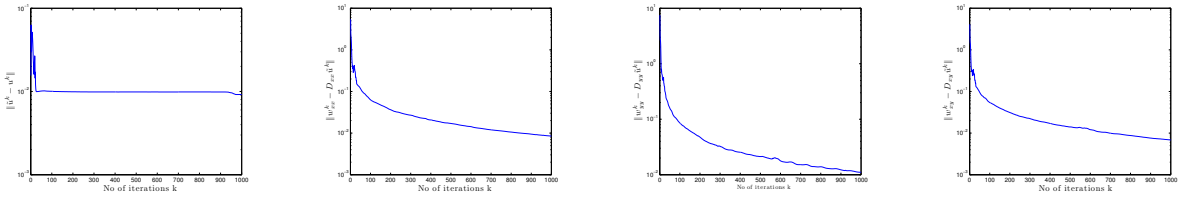


(d) $\|\tilde{u} - u\|_2$ (e) $\|w_{xx} - D_{xx}\tilde{u}\|_2$ (f) $\|w_{yy} - D_{yy}\tilde{u}\|_2$ (g) $\|w_{xy} - D_{xy}\tilde{u}\|_2$

Figure 17: $\beta = 0.001$, $\lambda_0 = 0.1$, $\lambda_2 = 0.01$. We have slow convergence, in 6680 iterations. The criterion (43) is satisfied in iteration 1150 but the inpainting domain is not filled in yet. The convergence is smooth, no oscillations are observed.

(a) 100 iterations (b) 500 iterations (c) 4230 iterations

(d) 100 iterations, details (e) 500 iterations, details (f) 4230 iterations, details



(g) $\|\tilde{u} - u\|_2$ (h) $\|w_{xx} - D_{xx}\tilde{u}\|_2$ (i) $\|w_{yy} - D_{yy}\tilde{u}\|_2$ (j) $\|w_{xy} - D_{xy}\tilde{u}\|_2$

Figure 18: $\beta = 0.001$, $\lambda_0 = 0.00001$, $\lambda_2 = 0.01$. In this case the inpainting domain is filled rather quickly and without any oscillations. However the continuation of the image in the boundary of the inpainting domain is not smooth in the first iterations but only after 4230 iterations, as figures (d), (e) and (f) reveal. The reason for that, is the fact that the constraint $u = \tilde{u}$ is taking a lot of iterations to be satisfied, see figure (g).

are observed, convergence is again dramatically slow, achieved in 6680 iterations. The constraints in (5) are satisfied but not to the same degree as in the examples of figures 14 and 15.

Finally in the last example, figure 18, keeping $\lambda_2$ fixed, we decrease the value of $\lambda_0$ down to $10^{-5}$. In this example the inpainting domain is filled in quickly and to the human eye it might seem that the method has converged very quickly, see for example figure 18(a) where the 100th iterate is shown. However, actual convergence is achieved in 4230 iterations. By looking at the zoom in details in figures 18(d), (e) and (f), the reader can convince himself that indeed the result after 100 iterations is not accurate. This can be explained by looking at figure 18(g) and observing that the constraint $u = \tilde{u}$ is not satisfied yet.

In conclusion, through the previous examples, it is justified that the choice of $\lambda$'s (44) leads to a fairly fast convergence. However further investigation is needed in order to examine the exact role that these parameters have in the convergence speed of the split Bregman iteration.

# References

[1] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*. Oxford University Press, USA, 2000. ISBN: 0198502451.

[2] P. Arias, V. Caselles, and G. Sapiro. A variational framework for non-local image inpainting. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 345–358. Springer, 2009. http://dx.doi.org/10.1007/978-3-642-03641-5_26.

[3] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000. http://dx.doi.org/10.1145/344779.344972.

[4] A. L. Bertozzi and J. B. Greer. Low-curvature image simplifiers: Global regularity of smooth solutions and Laplacian limiting schemes. *Communications on Pure and Applied Mathematics*, 57(6):764–790, 2004. http://dx.doi.org/10.1002/cpa.20019.

[5] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, 2007. http://dx.doi.org/10.1007/s10851-007-0017-6.

[6] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[7] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez. Geometrically guided exemplar-based inpainting. *SIAM Journal on Imaging Sciences*, 4(4):1143–1179, 2009. http://dx.doi.org/10.1137/110823572.

[8] T. Chan, S. Kang, and J. Shen. Euler's elastica and curvature-based inpainting. *SIAM Journal on Applied Mathematics*, 63(2):564–592, 2002. http://dx.doi.org/10.1137/S0036139901390088.

[9] T. Chan and J. Shen. Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001. http://dx.doi.org/10.1006/jvci.2001.0487.

[10] T. Chan and J. Shen. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002. http://www.jstor.org/stable/3061798.

[11] T. Chan and J. Shen. Variational image inpainting. *Communications on pure and applied mathematics*, 58(5):579–619, 2005. http://dx.doi.org/10.1002/cpa.20075.

[12] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–721, 2003. http://dx.doi.org/10.1109/TIP.2004.833105.

[13] F. Demengel. Fonctions à Hessien borné. *Annales de l'institut Fourier*, 34:155–190, 1985.

[14] S. Esedoglu and J. Shen. Digital inpainting based on the Mumford–Shah–Euler image model. *European Journal of Applied Mathematics*, 13(4):353–370, 2002. http://dx.doi.org/10.1017/S0956792502004904.

[15] E. Giusti. *Minimal surfaces and functions of bounded variation*, volume 80 of *Monographs in Mathematics*. Birkhäuser, Boston–Basel–Stuttgart, 1984. ISBN: 0817631534.

[16] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, 2:323, 2009. http://dx.doi.org/10.1137/080725891.

[17] R. Lai, X. Tai, and T. Chan. A ridge and corner preserving model for surface restoration. *UCLA CAM Report 11-55*, 2011. ftp://ftp.math.ucla.edu/pub/camreport/cam11-55.pdf.

[18] S. Masnou and J. Morel. Level lines based disocclusion. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 259–263. IEEE, 1998. http://dx.doi.org/10.1109/ICIP.1998.999016.

[19] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling and Simulation*, 4:460–489, 2005. http://dx.doi.org/10.1137/040605412.

[20] K. Papafitsoros and C. Schönlieb. A combined first and second order variational approach for image reconstruction. *arXiv:1202.6341v1, submitted*, 2012. http://arxiv.org/abs/1202.6341.

[21] Pascal Getreuer. Rudin-Osher-Fatemi Total Variation Denoising using Split Bregman. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.g-tvd.

[22] Pascal Getreuer. Total Variation Deconvolution using Split Bregman. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.g-tvdc.

[23] Pascal Getreuer. Total Variation Inpainting using Split Bregman. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.g-tvi.

[24] C. Schönlieb and A. Bertozzi. Unconditionally stable schemes for higher order inpainting. *Communications in Mathematical Sciences*, 9(2):413–457, 2011.

[25] J. Shen and T. Chan. Variational restoration of non-flat image features: models and algorithms. *SIAM Journal on Applied Mathematics*, 61:1338–1361, 2001. http://dx.doi.org/10.1137/S003613999935799X.

[26] X. Tai, J. Hahn, and G. Chung. A fast algorithm for Euler's elastica model using augmented Lagrangian method. *SIAM Journal on Imaging Sciences*, 4(1):313–344, 2011. http://dx.doi.org/10.1137/100803730.

[27] C. Wu and X. Tai. Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. *SIAM Journal on Imaging Sciences*, 3(3):300–339, 2010. http://dx.doi.org/10.1137/090767558.

[28] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for $\ell_1$-minimisation with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1:142–168, 2008. http://dx.doi.org/10.1137/070703983.