



# Learning Consistent Discretizations of the Total Variation

Antonin Chambolle, Thomas Pock

## ► To cite this version:

Antonin Chambolle, Thomas Pock. Learning Consistent Discretizations of the Total Variation. 2020.  
hal-02982082v1

**HAL Id: hal-02982082**

**<https://hal.science/hal-02982082v1>**

Preprint submitted on 28 Oct 2020 (v1), last revised 17 Sep 2021 (v2)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Learning Consistent Discretizations of the Total Variation\*

Antonin Chambolle<sup>†</sup>      Thomas Pock<sup>‡</sup>

October 27, 2020

In this work, we study a general framework of discrete approximations of the total variation for image reconstruction problems. The framework, for which we can show consistency in the sense of  $\Gamma$ -convergence, unifies and extends several existing discretization schemes. In addition, we propose algorithms for learning discretizations of the total variation in order to achieve the best possible reconstruction quality for particular image reconstruction tasks. Interestingly, the learned discretizations significantly differ between the tasks, illustrating that there is no universal best discretization of the total variation.

**Keywords:** Total variation, image denoising, image inpainting, discretization, finite differences, finite elements, learning, bi-level optimization, primal-dual algorithms.

**AMS MSC (2020):** 49Q20 35J87 35R35 49M29 94A08 65D18 65N06 65N30 65N50

## 1. Introduction

This paper tries to explore some questions raised in a previous work [10] (see also [4]), where the authors considered alternative discretizations on the total variation, based on finite elements, for image denoising or (basic) inpainting. In the latter experimental setting, the issue is to recover a missing discontinuity from its boundary datum, and it was observed that even when a discrete approximation is enjoying the property that the discretization of a sharp edge is, whatever the direction, a minimizer of the discrete inpainting problem (as in the case for the line labeled “RT” in Fig. 2, see [4] and [11, Prop. 4.1]), results could substantially differ in terms of sharpness and precision. In the series of experiments carried on in [10, 4], the best results for inpainting discontinuities

---

\*This work was funded by ERC grant HOMOVIS, No. 640156.

<sup>†</sup>CEREMADE, CNRS & Université Paris-Dauphine PSL, 75016 Paris, France / previously with CMAP, École Polytechnique, CNRS, Palaiseau, France ([chambolle@ceremade.dauphine.fr](mailto:chambolle@ceremade.dauphine.fr)).

<sup>‡</sup>Institute of Computer Graphics and Vision, Graz University of Technology, 8010 Graz, Austria ([pock@icg.tugraz.at](mailto:pock@icg.tugraz.at)).

were obtained with a discretization introduced in [17] and properly implemented in [12], called the “Condat” discretization. Mimicking the standard definition (by duality) of the total variation (here to simplify, on a bidimensional bounded open domain  $\Omega \subset \mathbb{R}^2$ ):

$$TV(u) := \sup \left\{ - \int_{\Omega} u \operatorname{div} \mathbf{p} \, dx : \mathbf{p} \in C_c^\infty(\Omega; \mathbb{R}^2), |\mathbf{p}(x)| \leq 1 \, \forall x \in \Omega \right\} \quad (1)$$

for a function  $u \in L^1(\Omega)$  (we address only the scalar problem, valid for instance for gray-level images), the idea of [17, 12] is to define a discrete total variation of a matrix of pixel values  $(u_{i,j})$  as  $\sup \sum_{i,j} p_{i+\frac{1}{2},j}^1 (u_{i+1,j} - u_{i,j}) + p_{i,j+\frac{1}{2}}^2 (u_{i,j+1} - u_{i,j})$ , with a particular set of constraints on the discrete dual variables  $\mathbf{p} = (p_{i+\frac{1}{2},j}^1, p_{i,j+\frac{1}{2}}^2)$ . While the usual forward-differences definition, such as considered in [5], simply requires the constraints  $(p_{i+\frac{1}{2},j}^1)^2 + (p_{i,j+\frac{1}{2}}^2)^2 \leq 1$  for all  $i, j$ , the idea of [17, 12] (see also the variants [14, 16]) is to linearly interpolate the discrete dual variables  $\mathbf{p} = (p_{i+\frac{1}{2},j}^1, p_{i,j+\frac{1}{2}}^2)$  into a continuous field  $\mathbf{p}(x) = (p^1(x), p^2(x))$  and enforce norm constraints for this continuous field in a few well chosen discrete locations.

A first question which is raised by the new definition in [17, 12] is whether it is consistent, that is, whether this it can be truly regarded as discretization of the total variation functional 1. Strangely, the proof is not as obvious as for more standard discretizations. The most rudimentary framework to assess this consistency is the so-called “ $\Gamma$ -convergence” [13, 3], a property which ensures that minimizers of the discrete total variation (plus various types of lower order terms) will converge, as the pixels’ size goes to zero, to a minimizer of the continuous one (1). This does not give any convergence rate or error bounds, but at least guarantees the consistency of the approach. Error bounds for variants have been discussed in various papers [24, 19, 10] (see also the review [11]) while in [4] the second author, together with Corentin Caillaud, report an essentially optimal error bound for the “Rudin-Osher-Fatemi” (that is, denoising) problem based on a variant of [17, 12] where the constraints on the dual fields  $\mathbf{p}(x)$  are enforced at each “corner” of a pixel, and which corresponds to considering in (1) a subset of the admissible fields, namely “Raviart-Thomas” dual fields (which are conforming finite element approximations of fields with divergence [23]) subject to the square pixels’ grid. Up to now, similar error bounds seem quite hard to establish for [17, 12], we observe however that both the latter and the generalizations we will consider here, obtained by minimizing some empirical error criterion, should in practice behave better than the variant studied in [4].

In this work, we start by addressing the consistency issue, in a more general framework. Our idea is that the previous approaches are easily generalized, by enforcing constraints on particular averages of the dual fields  $\mathbf{p}(x)$  (which might correspond to interpolants in particular points as in [4, 17, 12], but more generally are obtained by convolutions with kernels of small support). It follows a quite general definition, which also includes as a particular case the classical definition [5]. We show that this definition is consistent in the sense of  $\Gamma$ -convergence (Theorem 2.4).

In a next step, we address the issue of learning the parameters of the model, for different types of tasks. It means that we can try to find the “best” convolution kernels so that the denoising or inpainting problems are “best” solved for a given set of data.

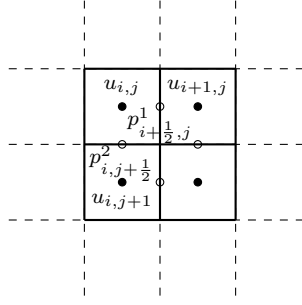


Figure 1: General setting of the discretization. Filled circles correspond to discrete pixel values  $u_{i,j}$  localized at the centers of the pixels. Unfilled circles correspond to the discrete dual variables, with the values  $p^1_{i+\frac{1}{2},j}$  localized on vertical edges and  $p^2_{i,j+\frac{1}{2}}$  localized on horizontal edges.

The idea is that (heuristically) the error of these kernels should always be better than the error bounds for the standard approaches which fit the framework. We show that such approach can lead to surprisingly good results for the given tasks, at the expense though of quite more complicated discrete total variations than the ones usually considered (as we replace the standard dual constraint by several — up to 8 in most of our examples and 40 for the denoising of natural images — constraints involving a whole neighborhood of pixels).

In a near future, we believe this framework can be adapted to address more complex energies of singular objects in 2D or 3D, and we plan to extend it for solving harder problems, such as the “total roto-translational variation” described in [9].

The plan of the paper is as follows: in the next Sections 2–2.3 we describe more precisely our general setting. Then in Section 2.4 we state and prove the consistency result. In the second part of the paper, starting with Section 3, we address the learning of the parameters. We describe our strategy and, in Section 4, we apply it to several tasks such as the “denoising” and “inpainting problems”, which we describe precisely in the corresponding sections. Eventually in the appendix, we (partially) justify the differentiation technique which we have used for optimizing the parameters of our models.

## 2. Problem and setting

In this section, we define the general discretization framework and show that several popular discretization schemes are particular instances of this class of discretizations.

An image  $u \in X \simeq \mathbb{R}^{M \times N}$  is a matrix  $u = (u_{i,j})_{1 \leq i \leq M, 1 \leq j \leq N}$  of pixel values (here to simplify we consider scalar images). We introduce the operator  $\mathbf{D} : X \rightarrow Y \simeq \mathbb{R}^{(M-1) \times N} \times \mathbb{R}^{M \times (N-1)}$  defined as usual as  $\mathbf{D}u = (D^1u, D^2u)$  with

$$\begin{cases} (D^1u)_{i+\frac{1}{2},j} = u_{i+1,j} - u_{i,j} & i = 1, \dots, M-1, j = 1, \dots, N; \\ (D^2u)_{i,j+\frac{1}{2}} = u_{i,j+1} - u_{i,j} & i = 1, \dots, M, j = 1, \dots, N-1; \end{cases} \quad (2)$$

A natural way to discretize the total variation consists in introducing an appropriate

norm  $\|\cdot\|_Y$  on  $Y$ , typically some sort of “2, 1” norm, and let  $\|\mathbf{D}u\|_Y$  be the approximate total variation.

The norm  $\|\mathbf{D}u\|_Y$  can of course be defined by duality as

$$\|\mathbf{D}u\|_Y = \sup \{ \langle \mathbf{p}, \mathbf{D}u \rangle_Y : \|\mathbf{p}\|_Y^* \leq 1 \}$$

(where  $\|\cdot\|_Y^*$  is the dual (or polar) norm of  $\|\cdot\|_Y$ , defined in theory on the dual  $Y^*$  which here is identified with  $Y$  through a Euclidean scalar product  $\langle \cdot, \cdot \rangle_Y$ ). As illustrated in Figure 1, the discrete image pixel values  $u_{i,j}$  can be thought as pixel-averaged values of an underlying continuous function  $u$ , and are localized in the center of the pixels. The discrete dual variables  $\mathbf{p} = (p^1, p^2)$  with values  $p_{i+\frac{1}{2},j}^1$  on vertical edges and  $p_{i,j+\frac{1}{2}}^2$  on horizontal edges can be interpreted as edge-averaged fluxes of an underlying continuous field  $\mathbf{p}(x)$ , localized in the center of the edges.

The most standard choice is, for  $p \in Y$ , to let:

$$\|\mathbf{p}\|_Y := \sum_{i=1}^M \sum_{j=1}^N \sqrt{(p_{i+\frac{1}{2},j}^1)^2 + (p_{i,j+\frac{1}{2}}^2)^2} \quad (3)$$

where by convention one sets in the norm  $p_{M+\frac{1}{2},j}^1 = p_{i,N+\frac{1}{2}}^2 = 0$  for all  $i, j$ . This is known not to give very precise results and suffer anisotropy issues, see [4] for a recent study. Then, an improvement over (3), suggested in [17, 12], consists in replacing  $\|\mathbf{D}u\|_Y$  with another norm defined by means of an averaging operator  $\mathbf{F} : Y \rightarrow Z$ , where  $Z$  is another Euclidean space, and constraining in the dual formulation above  $\mathbf{F}\mathbf{p}$  rather than  $\mathbf{p}$ :

$$TV(u) := \sup \{ \langle \mathbf{p}, \mathbf{D}u \rangle_Y : \|\mathbf{F}\mathbf{p}\|_Z^* \leq 1 \} \quad (4)$$

In practice, in this paper,  $\mathbf{F}$  is of convolution type and  $Z$  has more or less the same structure as  $Y$  (and in general is a copy of  $L$  folds of  $Y$ ,  $L \geq 1$ ). As considered in [4, Sec. 5], this point of view is particularly natural in the setting where (to simplify, as later in Section 2.4) we have  $\Omega$  a rectangle,  $\varepsilon > 0$  a small parameter with  $N \sim M \sim 1/\varepsilon$ , and  $u$  is identified to the piecewise constant “Q0” function

$$\sum_{i,j} u_{i,j} \chi_{(i\varepsilon-\varepsilon, i\varepsilon) \times (j\varepsilon-\varepsilon, j\varepsilon)}(x). \quad (5)$$

Then, extending the discrete variable  $\mathbf{p}$  as a Raviart-Thomas field [23]  $\mathbf{p}(x)$  subject to the square mesh  $\bigcup_{i,j} (i\varepsilon - \varepsilon, i\varepsilon) \times (j\varepsilon - \varepsilon, j\varepsilon)$ , and with vanishing flux across  $\partial\Omega$ , the scaled expression  $\varepsilon \langle \mathbf{p}, \mathbf{D}u \rangle_Y$  is precisely equal to  $-\int_{\Omega} \operatorname{div} \mathbf{p}(x) u(x) dx$ , and one expects to recover a sound approximation of (1) if the constraint  $\|\mathbf{F}\mathbf{p}\|_Z^* \leq 1$  in (4) is a good approximation of the dual constraint in (1) for the field  $\mathbf{p}(x)$ . (We observe also that the expression  $-\int_{\Omega} \operatorname{div} \mathbf{p}(x) u(x) dx$  is well defined for any  $u \in L^1(\Omega)$ , and depends only on the  $(L^2)$  projection of  $u$  on piecewise constant functions of the form (5), consisting in replacing  $u$  on each small pixel  $(i\varepsilon - \varepsilon, i\varepsilon) \times (j\varepsilon - \varepsilon, j\varepsilon)$  by its average on the pixel as already suggested.)

An obvious interesting case, described in [4] and which enters our general framework, is when  $\mathbf{F}$  is chosen in such a way that the discrete constraint in (4) is precisely equivalent

to  $|\mathbf{p}(x)| \leq 1$  a.e. in  $\Omega$  for the Raviart-Thomas field  $\mathbf{p}(x)$ , yielding a functional which clearly, for any  $u \in L^1(\Omega)$ , is below (1) and can be shown to be a good approximation of the total variation in terms of error bound. It corresponds to the setting described in (7)–(8) in the next Section 2.1; we refer to [4, Sec. 5] (*cf* also [14, 11]), for more details and precise definitions. One objective of this paper is to generalize both [4] and [12].

Observe that for  $\mathbf{r} \in Y$  one has:

$$\begin{aligned} \sup_{\|\mathbf{F}\mathbf{p}\|_Z^* \leq 1} \langle \mathbf{p}, \mathbf{r} \rangle_Y &= \sup_{\mathbf{p}} \inf_{\mathbf{q}} \langle \mathbf{p}, \mathbf{r} \rangle_Y - \langle \mathbf{q}, \mathbf{F}\mathbf{p} \rangle_Z + \|\mathbf{q}\|_Z \\ &\leq \inf_{\mathbf{q}} \sup_{\mathbf{p}} \|\mathbf{q}\|_Z + \langle \mathbf{p}, \mathbf{r} - \mathbf{F}^*\mathbf{q} \rangle_Y = \inf_{\mathbf{q}: \mathbf{F}^*\mathbf{q}=\mathbf{r}} \|\mathbf{q}\|_Z \end{aligned}$$

(which is  $+\infty$  if the constraint is not feasible, that is, if  $\mathbf{r} \notin \text{Im}\mathbf{F}^*$ ). It is standard that this is an equality (using that the RHS is lower semicontinuous and its convex conjugate is precisely the characteristic of  $\{\mathbf{p} : \|\mathbf{F}\mathbf{p}\|_Z^* \leq 1\}$ ), provided of course the value of the last inf is  $+\infty$  when the feasible set is empty, which is when  $\mathbf{r} \notin \text{im}\mathbf{F}^* = (\ker \mathbf{F})^\perp$ .

Hence the discrete total variation (4) also has the primal form

$$TV(u) = \min_{\mathbf{q}: \mathbf{F}^*\mathbf{q}=\mathbf{D}u} \|\mathbf{q}\|_Z. \quad (6)$$

Observe that this formulation of the total variation can also be interpreted as a particular form of a “group-lasso” problem, where one seeks for a sparse representation  $\mathbf{F}^*\mathbf{q}$  of the discrete gradient  $\mathbf{D}u$  based on a convolutional dictionary  $\mathbf{F}^*$  [8]. We believe, that in the spirit of compressed sensing, this formulation enables the recovery of sharp discontinuities in discrete images and breaks the intrinsic band limitations of more classical, digital signal processing inspired discretizations, for example the Shannon total variation proposed in [1].

## 2.1. Some examples

In the following we mention a few popular discretizations which can all be seen as particular instances of our general discretization framework.

*Example 2.1* (Forward differences (FD)). Certainly, the most widely used discretization of the total variation is based on simple forward differences (see for example [5]). In our framework, this corresponds to defining  $\mathbf{F}\mathbf{p} \in Z \simeq Y$  as

$$(\mathbf{F}\mathbf{p})_{i,j} = \begin{pmatrix} p_{i+\frac{1}{2},j}^1 \\ p_{i,j+\frac{1}{2}}^2 \end{pmatrix}.$$

The corresponding  $Z$ -norm is then equivalent to (3). As the operator  $\mathbf{F}$  combines dual variables defined at different spatial locations, it follows that the constraint  $\|\mathbf{F}\mathbf{p}\|_Z^* \leq 1$  leads to a bias towards certain directions of the image gradient. This can be clearly observed in Figure 2, where some directions appear sharp, while others are overly blurred.

*Example 2.2* (Raviart-Thomas discretization of the dual (RT) [14, 4, 11]). In order to gain more rotational invariance, one can consider four constraints on the dual variables

at the four corners of each pixels. In our framework, this corresponds to letting  $\mathbf{p} \in Y$ ,  $\mathbf{F}\mathbf{p} = (F^1\mathbf{p}, F^2\mathbf{p}, F^3\mathbf{p}, F^4\mathbf{p}) \in Z$  and defining

$$\begin{aligned} (F^1\mathbf{p})_{i-\frac{1}{2},j-\frac{1}{2}} &= \begin{pmatrix} p_{i-\frac{1}{2},j}^1 \\ p_{i,j-\frac{1}{2}}^2 \end{pmatrix}, & (F^2\mathbf{p})_{i-\frac{1}{2},j+\frac{1}{2}} &= \begin{pmatrix} p_{i-\frac{1}{2},j}^1 \\ p_{i,j+\frac{1}{2}}^2 \end{pmatrix}, \\ (F^3\mathbf{p})_{i+\frac{1}{2},j-\frac{1}{2}} &= \begin{pmatrix} p_{i+\frac{1}{2},j}^1 \\ p_{i,j-\frac{1}{2}}^2 \end{pmatrix}, & (F^4\mathbf{p})_{i+\frac{1}{2},j+\frac{1}{2}} &= \begin{pmatrix} p_{i+\frac{1}{2},j}^1 \\ p_{i,j+\frac{1}{2}}^2 \end{pmatrix}. \end{aligned} \quad (7)$$

The corresponding  $Z$  norm is given by

$$\|(\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3, \mathbf{z}^4)\|_Z := \sum_{i,j} |\mathbf{z}_{i-\frac{1}{2},j-\frac{1}{2}}^1|_2 + |\mathbf{z}_{i-\frac{1}{2},j+\frac{1}{2}}^2|_2 + |\mathbf{z}_{i+\frac{1}{2},j-\frac{1}{2}}^3|_2 + |\mathbf{z}_{i+\frac{1}{2},j+\frac{1}{2}}^4|_2, \quad (8)$$

where  $|\cdot|_2$  is the standard Euclidean length. The constraint  $\|\mathbf{F}\mathbf{p}\|_Z^* \leq 1$  then combines all possible pairs of edges that meet in one of the four corners of each pixel. With this, one ensures feasibility of the Raviart-Thomas interpretation of the dual field as a dual variable in (1), as mentioned in the previous section. However, as seen on Figure 2, these constraints seem not appropriate for the resolution of edge inpainting problems, as they lead to a strong blur of the inpainted edges with non grid-aligned orientations. This is a bit puzzling, as one can show (in a more precise setting though than the experiments carried on in this paper) that straight line inpainting problems can be solved exactly by the discrete projection of the continuous solution with this particular discretization, see [11, Prop. 4.1]. It is likely that, as is the case for the variant studied in [10], such problem have non unique solutions and the optimization, or variants in the setting, favor blurry results over sharper ones.

*Example 2.3* (Condat's discretization (CD) [17, 12]). A more flexible discretization is obtained by linear interpolation of the dual variables at the three grid positions  $(i, j)$ ,  $(i + \frac{1}{2}, j)$ , and  $(i, j + \frac{1}{2})$ . This corresponds to letting  $\mathbf{p} \in Y$ ,  $\mathbf{F}\mathbf{p} = (F^1\mathbf{p}, F^2\mathbf{p}, F^3\mathbf{p}) \in Z$  with, for  $i = 1, \dots, M$ ,  $j = 1, \dots, N$ ,

$$(F^1\mathbf{p})_{i,j} = \begin{pmatrix} \frac{p_{i-\frac{1}{2},j}^1 + p_{i+\frac{1}{2},j}^1}{2} \\ \frac{p_{i,j-\frac{1}{2}}^2 + p_{i,j+\frac{1}{2}}^2}{2} \end{pmatrix},$$

(again the convention is that  $p = 0$  when the indices are out of range), and, for  $i \leq M-1$ ,

$$(F^2\mathbf{p})_{i+\frac{1}{2},j} = \begin{pmatrix} p_{i+\frac{1}{2},j}^1 \\ \frac{p_{i,j-\frac{1}{2}}^2 + p_{i,j+\frac{1}{2}}^2 + p_{i+1,j-\frac{1}{2}}^2 + p_{i+1,j+\frac{1}{2}}^2}{4} \end{pmatrix},$$

and for  $j \leq N-1$ :

$$(F^3\mathbf{p})_{i,j+\frac{1}{2}} = \begin{pmatrix} \frac{p_{i-\frac{1}{2},j}^1 + p_{i+\frac{1}{2},j}^1 + p_{i-\frac{1}{2},j+1}^1 + p_{i+\frac{1}{2},j+1}^1}{4} \\ p_{i,j+\frac{1}{2}}^2 \end{pmatrix}.$$

The  $Z$  norm is then  $\|(\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3)\|_Z := \sum_{i,j} |\mathbf{z}_{i,j}^1|_2 + |\mathbf{z}_{i+\frac{1}{2},j}^2|_2 + |\mathbf{z}_{i,j+\frac{1}{2}}^3|_2$ . In contrast to RT, the constraint  $\|\mathbf{F}\mathbf{p}\|_Z^* \leq 1$  in CD do not guarantee the feasibility of the Raviart-Thomas extension of the discrete dual field, yet it seems this is necessary to better discriminate sharper edges from smoother ones. And indeed, Figure 2 confirms the excellent result for inpainting straight edges with various orientations.

Table 1 visualizes the filter coefficients of the aforementioned handcrafted methods. Observe that Condat’s discretization (CD) is the only one which relies on some averaging of the dual variables.



















FD	CD				RT			
								
								

Table 1: Visualization of the filters corresponding to the different handcrafted discretization methods. The filter sizes are  $2 \times 3$  for the horizontal direction and  $3 \times 2$  for the vertical direction, which corresponds to a support of  $2 \times 2$  pixels. Black corresponds to filters weights of 0.0, white corresponds to filter weights of 1.0, gray corresponds to filter weights of 0.5 and dark gray corresponds to filter weights of 0.25.

## 2.2. General interpolation operators

In practice, as in the examples above, we consider general convolution-type operators  $\mathbf{F}$ . The basic form for such  $\mathbf{F}$  is  $\mathbf{F} = (\mathbf{F}^l)_{l=1}^L$  with

$$(\mathbf{F}^l \mathbf{p})_{i,j} = \begin{pmatrix} (F^{l,1} p^1)_{i,j} \\ (F^{l,2} p^2)_{i,j} \end{pmatrix} = \begin{pmatrix} \sum_{m,n=-\nu}^{\nu} \xi_{m,n}^l p_{i+\frac{1}{2}-m, j-n}^1 \\ \sum_{m,n=-\nu}^{\nu} \eta_{m,n}^l p_{i-m, j+\frac{1}{2}-n}^2 \end{pmatrix}, \quad (9)$$

where  $(\xi_{m,n}^l, \eta_{m,n}^l)$  are weights with a small support of maximal size  $(2\nu + 1) \times (2\nu + 1)$  for some integer  $\nu \geq 0$ <sup>1</sup>. On order to ensure that  $F^{l,1}, F^{l,2}$  represent valid interpolation kernels, we shall assume that the entries of the filters sum up to one, that is,

$$\sum_{m,n} \xi_{m,n}^l = \sum_{m,n} \eta_{m,n}^l = 1 \iff F^{l,1}, F^{l,2} \in C_{\Sigma=1}, \quad (10)$$

where here  $C_{\Sigma=1}$  denotes the set of filters whose coefficients sum up to one. It is not necessary to assume that the weights are non-negative, but if not we must assume they are globally bounded (in absolute value) by some fixed constant. A generic point in  $Z$  has the form  $\mathbf{q} = (\mathbf{q}_{i,j}^1, \dots, \mathbf{q}_{i,j}^L)$ , where each  $(\mathbf{q}_{i,j}^l)$  is a 2-dimensional vector, for indices

<sup>1</sup>In practice, we will consider rectangular filters, but they can be always implemented by means of larger square filters and setting the respective filter weights to zero.



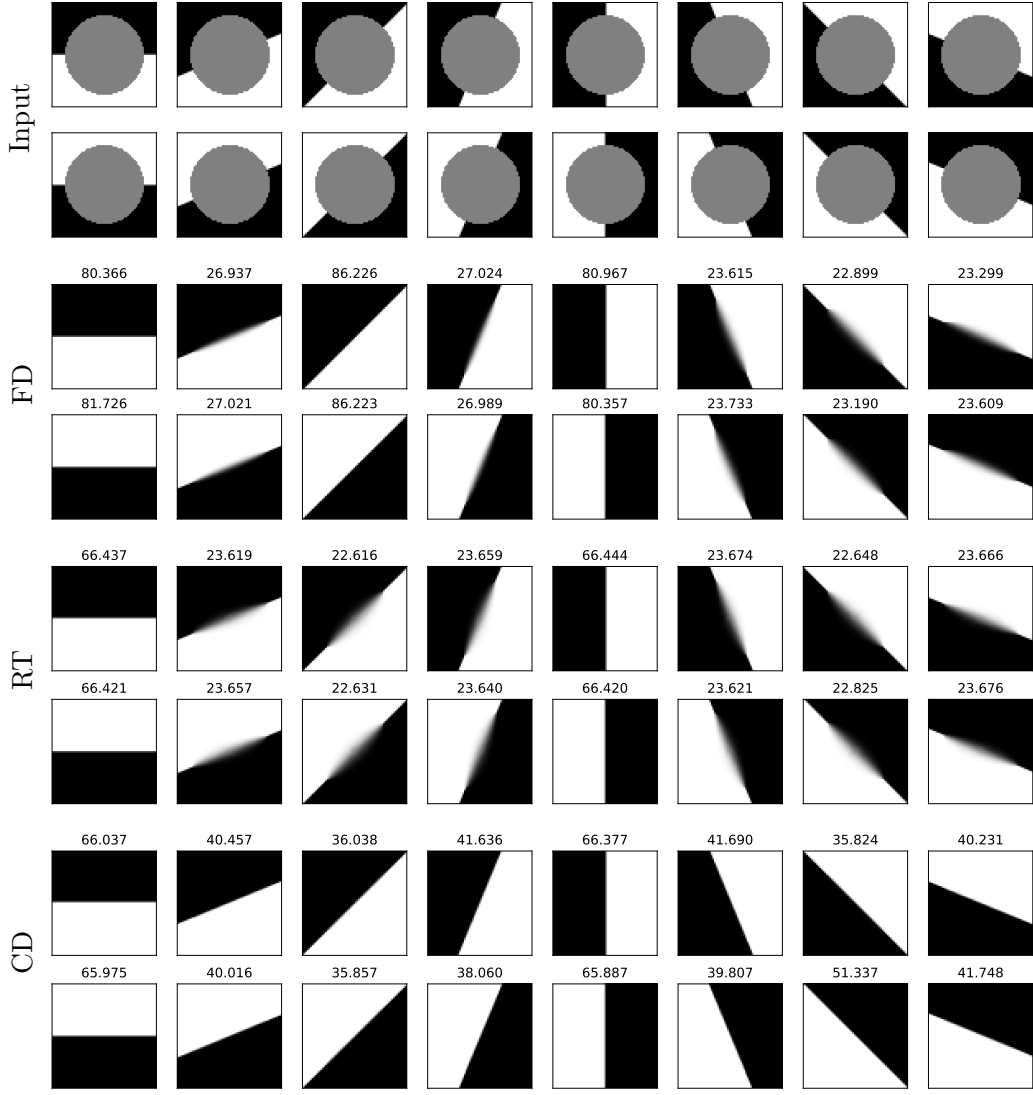


Figure 2: Reconstruction quality of various “handcrafted” discretization schemes for the problem of inpainting a straight discontinuity. The numbers above the images refer to the reconstruction errors, measured in PSNR.

$i, j \in \mathbb{Z}^2$ ; relevant indices are in  $\{1, \dots, M\} \times \{1, \dots, N\}$  plus the support of the convolution kernels  $(\xi^l, \eta^l)$ . Then, the norms we consider are

$$\|\mathbf{q}\|_Z = \sum_{i,j,l} |\mathbf{q}_{i,j}^l|_2, \quad \|\mathbf{q}\|_Z^* = \max_{i,j,l} |\mathbf{q}_{i,j}^l|_2.$$

The constraint in the primal problem  $\mathbf{F}^* \mathbf{q} = \mathbf{D}u$  reads (denoting  $\mathbf{q}_{i,j}^l = (q_{i,j}^{l,1}, q_{i,j}^{l,2})^T$ ):

$$(D^1 u)_{i+\frac{1}{2},j} = \sum_{m,n,l} \xi_{m,n}^l q_{i+m,j+n}^{l,1}, \quad (D^2 u)_{i,j+\frac{1}{2}} = \sum_{m,n,l} \eta_{m,n}^l q_{i+m,j+n}^{l,2}.$$

### 2.3. Boundary conditions

In this section we briefly discuss how the most popular boundary conditions can be realized in our framework.

**Homogeneous Neumann conditions** Homogeneous Neumann conditions are treated as follows: we assume that  $\mathbf{p} \in Y$ , that is, the values of  $\mathbf{p}$  are defined exactly at the same points as the values of  $D^1u, D^2u$ . For computing the convolutions, we implicitly assume  $p = 0$  when the indices fall out of range. The variable  $\mathbf{q} \in Z$ , on the other hand, should be nonzero for a larger set of indices, namely  $i \in \{-\nu, \dots, M + \nu\}, j \in \{-\nu, \dots, N + \nu\}$ .

**Homogeneous Dirichlet conditions** For Dirichlet conditions, it is roughly the opposite. Indeed, one can assume that  $u = 0$  and  $D^\alpha u = 0, \alpha = 1, 2$ , out of the domain, that is,  $u_{i,j} = 0$  if  $i \leq 0$  or  $j \leq 0$  or  $i \geq M + 1$  or  $j \geq N + 1$ . In particular, the variable  $\mathbf{p}$  multiplying a difference  $Du$  which is zero should be unconstrained, *i.e.*, for instance,  $p_{i+\frac{1}{2},0}^1$  for any  $i$ . Then, one should consider only convolutions  $\mathbf{F}^*\mathbf{p}$  which do not involve unconstrained values of  $\mathbf{p}$ . This amounts to let  $q_{i,j}^{l,\alpha} = 0$ , when the convolution  $(F^{l,\alpha}p^\alpha)_{i,j}$  involves values of  $\mathbf{p}$  out of the support (*cf.* (9)), in particular this implies that  $(F^{l,\alpha})^*q^{l,\alpha} = 0$  at points where  $D^\alpha u$  has been forced to zero, in coherence with the constraint  $\mathbf{F}^*\mathbf{q} = Du$ .

**Heterogeneous Dirichlet conditions** In practice, for solving problems with non homogeneous boundary conditions, we will consider the Neumann discretization introduced above, and will solve our problems with an additional equality constraint on  $u$  in a strip of width the size of the convolution operators.

### 2.4. General form: consistency

To simplify in this section we let  $N = M$ , and consider the domain  $\Omega = (0, 1)^2$ , divided in  $N \times N$  pixels of size  $\varepsilon \times \varepsilon, \varepsilon = 1/N$ . We add the subscript  $\varepsilon$  to all the notions introduced in the previous section, which now depend on  $N = 1/\varepsilon$ , which will be sent to  $\infty$ . We let  $\mathbf{D}_\varepsilon = (1/\varepsilon)\mathbf{D} : X_\varepsilon \rightarrow Y_\varepsilon, \mathbf{D}$  defined as in (2). We define

$$TV_\varepsilon(u) = \min \left\{ \varepsilon^2 \|\mathbf{q}\|_{Z_\varepsilon} : \mathbf{F}_\varepsilon^* \mathbf{q} = \mathbf{D}_\varepsilon u \right\} = \sup \left\{ \varepsilon^2 \langle \mathbf{p}, \mathbf{D}_\varepsilon u \rangle_{Y_\varepsilon} : \|\mathbf{F}_\varepsilon \mathbf{p}\|_{Z}^* \leq 1 \right\}$$

when  $u \in L^1(\Omega)$  is of the form  $u = \sum_{i,j=1}^N u_{i,j} \chi_{(i\varepsilon-\varepsilon, i\varepsilon) \times (j\varepsilon-\varepsilon, j\varepsilon)}$ , and  $+\infty$  else. We only consider, for simplification, the case of Neumann homogeneous boundary conditions as described in the previous section. Then the following holds:

**Theorem 2.4.** *Assume the supports and the weights of the convolutions defining  $\mathbf{F}_\varepsilon$  are uniformly bounded. Then  $TV_\varepsilon$   $\Gamma$ -converges to*

$$TV(u) := \begin{cases} |Du|(\Omega) & \text{if } u \in BV(\Omega), \\ +\infty & \text{else.} \end{cases}$$

The convergence both holds in very weak topologies (such as distributional), and in stronger ones such as  $L^p(\Omega)$  for any  $p < +\infty$ .

*Proof.* To prove the “lim inf”, we assume  $u^\varepsilon \rightarrow u$  in the distributional sense and that  $\liminf_\varepsilon TV_\varepsilon(u^\varepsilon)$  is finite. Consider  $\mathbf{p}(x) = (p^1(x), p^2(x)) \in C_c^\infty(\Omega; \mathbb{R}^2)$  a smooth vector field with compact support and  $|\mathbf{p}(x)| \leq 1$  for every  $x \in \Omega$ . Then

$$\int_\Omega u \operatorname{div} \mathbf{p} dx = \lim_{\varepsilon \rightarrow 0} \int_\Omega u^\varepsilon \operatorname{div} \mathbf{p} dx = \lim_{\varepsilon \rightarrow 0} \varepsilon^2 \langle \mathbf{D}_\varepsilon u^\varepsilon, \mathbf{p}^\varepsilon \rangle_{Y_\varepsilon}$$

where the discretized  $\mathbf{p}^\varepsilon$  is given by the horizontal and vertical fluxes<sup>2</sup>

$$p_{i+\frac{1}{2},j}^{\varepsilon,1} = (1/\varepsilon) \int_{(j-1)\varepsilon}^{j\varepsilon} p^1(i\varepsilon, y) dy, \quad p_{i,j+\frac{1}{2}}^{\varepsilon,2} = (1/\varepsilon) \int_{(i-1)\varepsilon}^{i\varepsilon} p^2(x, j\varepsilon) dx.$$

Since  $\mathbf{p}$  is smooth, there exists  $C$  depending only on  $\|\nabla \mathbf{p}\|_\infty$  and  $\mathbf{F}$  such that  $|(F^l \mathbf{p}^\varepsilon)_{i,j}| \leq 1 + C\varepsilon$  for all  $i, j, l$ . Indeed, using the form (9), one sees that, denoting  $x_{i,j}^\varepsilon := ((i + \frac{1}{2})\varepsilon, (j + \frac{1}{2})\varepsilon)$  the center point of the pixel  $(i\varepsilon, (i+1)\varepsilon) \times (j\varepsilon, (j+1)\varepsilon)$ ,

$$(F^{l,1} p^{\varepsilon,1})_{i,j} = \sum_{m,n=-\nu}^{\nu} \xi_{m,n}^l p_{i+\frac{1}{2}-m, j-n}^{\varepsilon,1} = p^1(x_{i,j}^\varepsilon) + \sum_{m,n=-\nu}^{\nu} \xi_{m,n}^l (p_{i+\frac{1}{2}-m, j-n}^{\varepsilon,1} - p^1(x_{i,j}^\varepsilon))$$

Being  $\mathbf{p}$  smooth, the last term can be bounded by a quantity  $C\varepsilon/\sqrt{2}$ , for some constant  $C \sim \nu \|\nabla \mathbf{p}\|_\infty \sum_{m,n} |\xi_{m,n}^l|$  which, by assumption, is uniformly bounded. In the same way,  $|(F^{l,2} p^{\varepsilon,2})_{i,j} - p^2(x_{i,j}^\varepsilon)| \leq C\varepsilon/\sqrt{2}$ . Since  $|\mathbf{p}(x_{i,j}^\varepsilon)|_2 \leq 1$ , our claim follows.

We deduce that  $\|\mathbf{F} \mathbf{p}^\varepsilon\|_{\mathbb{Z}}^* \leq 1 + C\varepsilon$ , so that  $\mathbf{p}^\varepsilon/(1 + C\varepsilon)$  is an admissible dual variable and  $\varepsilon^2 \langle \mathbf{D}_\varepsilon u^\varepsilon, \mathbf{p}^\varepsilon \rangle_{Y_\varepsilon} \leq (1 + C\varepsilon) TV_\varepsilon(u^\varepsilon)$ . Sending  $\varepsilon$  to zero and then taking the supremum with respect to all  $\mathbf{p}$  we obtain that:

$$TV(u) \leq \liminf_{\varepsilon \rightarrow 0} TV_\varepsilon(u^\varepsilon). \quad (11)$$

We remark that near the boundaries,  $\mathbf{p}^\varepsilon$  vanishes, so this construction remains valid in the framework of the Neumann boundary conditions described in Section 2.3.

Let us now prove the “lim sup”. Let us consider  $u \in BV(\Omega)$ . We need to build a “recovery” sequence, that is a sequence  $(u^\varepsilon)$  of discrete functions of the form

$$u^\varepsilon(x) = \sum_{i,j=1}^N u_{i,j}^\varepsilon \chi_{(i\varepsilon-\varepsilon, i\varepsilon) \times (j\varepsilon-\varepsilon, j\varepsilon)}(x), \quad (12)$$

for  $\varepsilon = 1/N$ ,  $N \geq 1$ , such that  $u^\varepsilon \rightarrow u$  in  $L^1(\Omega)$  (or actually in some  $L^p(\Omega)$ ,  $1 \leq p < \infty$ , if  $u \in L^p(\Omega)^3$ ) and:

$$\limsup_{\varepsilon \rightarrow 0} TV_\varepsilon(u^\varepsilon) \leq TV(u). \quad (13)$$

As is classical, a first observation is that one may assume that  $u$  is bounded, indeed the sequence  $u_k := -k \vee (u \wedge k)$  satisfies  $u_k \rightarrow u$  in  $L^1(\Omega)$  and  $\int_\Omega |Du_k| \rightarrow \int_\Omega |Du|$ . Then, from a recovery sequence for each  $u_k$ , one easily builds using a diagonal argument a recovery sequence for  $u$ .

<sup>2</sup>The notation might seem a bit inconsistent, as the centers of the pixels correspond in fact to the points  $x_{i,j}^\varepsilon := ((i + \frac{1}{2})\varepsilon, (j + \frac{1}{2})\varepsilon)$  while we choose to denote, as is usual, by  $(i, j)$  this location.

<sup>3</sup>To simplify we consider in the sequel only the case  $p = 1$ .

In a first step, as  $\Omega = (0, 1)^2$  is a square (a rectangle would work as well, for a general domain one should use a more complicated approach), it is clear that one can consider  $u$  as an even, 2-periodic functions (first let  $u(-x, y) = u(x, -y) = u(-x, -y) := u(x, y)$  for  $(x, y) \in (0, 1)^2$ , then extend  $u$  2-periodically in both directions). Observe that the Fourier transform of  $u$ , in that case, is real, even, and discrete. We can let, for  $\ell \in \mathbb{Z}^2$ ,

$$\hat{u}(\ell) := \frac{1}{4} \int_{[-1, 1]^2} u(x) e^{-2i\pi \frac{\ell \cdot x}{2}} dx = \int_{[0, 1]^2} u(x) \cos(\pi \ell \cdot x) dx$$

so that  $u(x) = \sum_{\ell \in \mathbb{Z}^2} \hat{u}(\ell) e^{i\pi \ell \cdot x}$ . By standard mollification (with a nonnegative, smooth and symmetric averaging kernel) one can assume also that  $u \in C^\infty(\mathbb{R}^2)$  (as one can approximate  $u$  with such functions, with again convergence of the total variations as the smoothing goes to zero).

Then, we consider  $\hat{\rho} \in C_c^\infty(B(0, 1); [0, 1])$  with  $\hat{\rho} \equiv 1$  on  $B(0, 1/2)$  a smooth, symmetric cutoff function (in the Fourier domain), and let for  $\ell \in \mathbb{Z}^2$ ,  $\hat{u}_\delta(\ell) := \hat{u}(\ell) |\hat{\rho}(\delta \ell / 2)|^2$ . It turns out that this is equivalent, in the spatial domain, to convolving  $u$  with  $\rho_\delta * \rho_\delta$ , where  $\rho_\delta(x) := (1/\delta^2) \rho(x/\delta)$  (is a smoothing kernel which is however not non-negative), and one has  $u_\delta \rightarrow u$  in  $L^1(\Omega)$  with  $TV(u_\delta) \rightarrow TV(u)$ . The latter holds for instance because as  $u$  was initially smooth we had in particular  $\int_{[0, 1]^2} |\nabla u|^2 dx < \infty$ , and passing to the Fourier domain it gets obvious that  $\nabla u_\delta \rightarrow \nabla u$  strongly in  $L^2(\Omega)$ , therefore also in  $L^1(\Omega)$ .

Hence, again, without loss of generality, one needs just to build a recovery sequence for a function  $u$  (real, even, 2-periodic) with vanishing spectrum out of a disk:  $\hat{u}(\ell) = 0$  if  $|\ell| \geq 2/\delta =: R$ , for some fixed value of  $\delta$ .

Given  $N > R$  an integer and  $\varepsilon = 1/N$ , let, for  $(i, j) \in \mathbb{Z}^2$

$$u_{i,j}^\varepsilon = u(i\varepsilon, j\varepsilon) = \sum_{n,m=-N}^N \hat{u}(n, m) e^{i\pi \frac{in+jm}{N}}. \quad (14)$$

This is of course real, even and periodic ( $u_{i+2N,j}^\varepsilon = u_{i,j+2N}^\varepsilon = u_{i,j}^\varepsilon$  for all  $i, j$ ), and it is natural to view the restriction  $(u_{i,j}^\varepsilon)_{1 \leq i,j \leq N}$  and its piecewise constant extension (12) as a discrete approximation of the original image in the domain  $\Omega = (0, 1)^2$ .

In addition, being  $u$  (very) smooth, it is obvious that  $u^\varepsilon$  (defined by (12)-(14)) converges in  $L^1(\Omega)$  (and in fact uniformly) to  $u$ , in the same way, as  $\varepsilon \rightarrow 0$  ( $N \rightarrow \infty$ ),

$$\sum_{i,j=1}^N \frac{u_{i,j}^\varepsilon - u_{i-1,j}^\varepsilon}{\varepsilon} \chi_{(i\varepsilon-\varepsilon, i\varepsilon) \times (j\varepsilon-\varepsilon, j\varepsilon)}(x) \rightarrow \partial_1 u(x)$$

(also uniformly) and in particular

$$\limsup_{\varepsilon \rightarrow 0} \varepsilon \sum_{i,j=1}^N \sqrt{(u_{i,j}^\varepsilon - u_{i-1,j}^\varepsilon)^2 + (u_{i,j}^\varepsilon - u_{i,j-1}^\varepsilon)^2} \leq \int_{\Omega} |\nabla u| dx = TV(u).$$

Let us now estimate  $TV_\varepsilon(u^\varepsilon)$ . We use the definition:

$$TV_\varepsilon(u^\varepsilon) = \varepsilon^2 \min \{ \|q\|_{Z_\varepsilon} : F_\varepsilon^* q = D_\varepsilon u^\varepsilon \}$$

with  $\mathbf{F}_\varepsilon$  a sum of  $L$  convolution operators as in (9). We need to find a  $\mathbf{q}$  such that<sup>4</sup>

$$\varepsilon^2 \|\mathbf{q}\|_{Z_\varepsilon} \lesssim \varepsilon \sum_{i,j=1}^N \sqrt{(u_{i,j}^\varepsilon - u_{i-1,j}^\varepsilon)^2 + (u_{i,j}^\varepsilon - u_{i,j-1}^\varepsilon)^2}. \quad (15)$$

In particular, it is enough to find  $\mathbf{q}$  of the form  $(\mathbf{q}^1, 0, \dots, 0)$ , so that without loss of generality, one can assume  $L = 1$  and drop the superscript  $l$ . Therefore, we look for  $\mathbf{q} = (\mathbf{q}_{i,j})$  ( $i, j \in \mathbb{Z}$ ,  $\mathbf{q}_{i,j} = (q_{i,j}^1, q_{i,j}^2)^T \in \mathbb{R}^2$  for all  $i, j$ ) which satisfies

$$(D_\varepsilon^1 u^\varepsilon)_{i+\frac{1}{2},j} = \sum_{m,n=-\nu}^{\nu} \xi_{m,n} q_{i+m,j+n}^1, \quad (D_\varepsilon^2 u^\varepsilon)_{i,j+\frac{1}{2}} = \sum_{m,n=-\nu}^{\nu} \eta_{m,n} q_{i+m,j+n}^2 \quad (16)$$

at least for  $1 \leq i \leq N-1$ ,  $1 \leq j \leq N$  for the first equality and  $1 \leq i \leq N$ ,  $1 \leq j \leq N-1$  for the second, and hoping to have a control on the norm  $\|\mathbf{q}\|_Z$ .

We compute the discrete Fourier transform of the first equation: for  $(r, s) \in \mathbb{Z}^2$ , one has

$$\begin{aligned} \frac{1}{2N} \sum_{i,j=-N}^N \frac{u_{i+1,j}^\varepsilon - u_{i,j}^\varepsilon}{\varepsilon} e^{-i\pi \frac{ir+js}{N}} &= \frac{1}{2N} \sum_{i,j=-N}^N \sum_{m,n=-\nu}^{\nu} \xi_{m,n} q_{i+m,j+n}^1 e^{-i\pi \frac{ir+js}{N}} \\ &= \frac{1}{2N} \sum_{i=-N}^N \sum_{m,n=-\nu}^{\nu} \xi_{m,n} e^{i\pi \frac{mr+ns}{N}} q_{i+m,j+n}^1 e^{-i\pi \frac{(i+m)r+(j+n)s}{N}} \\ &= \hat{q}^1(r, s) \left( \sum_{m,n=-\nu}^{\nu} \xi_{m,n} e^{i\pi \frac{mr+ns}{N}} \right). \end{aligned}$$

On the other hand,

$$\frac{1}{2N} \sum_{i,j=-N}^N \frac{u_{i+1,j}^\varepsilon - u_{i,j}^\varepsilon}{\varepsilon} e^{-i\pi \frac{ir+js}{N}} = \frac{e^{i\pi r\varepsilon} - 1}{\varepsilon} \hat{u}(r, s) \quad (17)$$

vanishes for  $\|(r, s)\| > R$ . Using that  $\sum_{m,n} \xi_{m,n} = 1$  one sees that

$$\sum_{m,n=-\nu}^{\nu} \xi_{m,n} e^{i\pi \frac{mr+ns}{N}} = 1 + \sum_{m,n=-\nu}^{\nu} \xi_{m,n} (e^{i\pi \frac{mr+ns}{N}} - 1) \rightarrow 1$$

as  $N \rightarrow \infty$  for any  $r, s$  with  $\|(r, s)\| \leq R$ . In particular, for  $N$  large enough and  $\|(r, s)\| \leq R$ , one can just let

$$\hat{q}^1(r, s) = \frac{\frac{e^{i\pi r\varepsilon} - 1}{\varepsilon} \hat{u}(r, s)}{1 + \sum_{m,n=-\nu}^{\nu} \xi_{m,n} (e^{i\pi \frac{mr+ns}{N}} - 1)}$$

(and one lets  $\hat{q}^1(r, s) = 0$  for  $\|(r, s)\| > R$ ). This choices guarantees that the first equation in (16) is satisfied, and one does analogously for the second. One could fear

---

<sup>4</sup>to simplify the notation we do not denote the dependency of  $\mathbf{q}$  on  $\varepsilon$ .

that the resulting  $(q_{i,j})$  is a complex vector field, yet since also its real part then should satisfy (16), and have therefore the same Fourier transform, this cannot be the case.

It remains to show that with this choice,  $\|q\|_{Z_\varepsilon}$  satisfies (15). One has

$$q_{i,j}^1 = \sum_{r,s=-N}^N \hat{q}^1(r,s) e^{i\pi \frac{ir+js}{N}} = \sum_{\|(r,s)\| \leq R} \frac{\frac{e^{i\pi r\varepsilon} - 1}{\varepsilon} \hat{u}(r,s)}{1 + \sum_{m,n=-\nu}^{\nu} \xi_{m,n} (e^{i\pi \frac{mr+ns}{N}} - 1)} e^{i\pi \frac{ir+js}{N}}$$

So that (using (17)),

$$(D_\varepsilon^1 u^\varepsilon)_{i+\frac{1}{2},j} - q_{i,j}^1 = \sum_{\|(r,s)\| \leq R} \frac{\sum_{m,n=-\nu}^{\nu} \xi_{m,n} (e^{i\pi \frac{mr+ns}{N}} - 1)}{1 + \sum_{m,n=-\nu}^{\nu} \xi_{m,n} (e^{i\pi \frac{mr+ns}{N}} - 1)} \frac{e^{i\pi r\varepsilon} - 1}{\varepsilon} \hat{u}(r,s) e^{i\pi \frac{ir+js}{N}}$$

and

$$|(D_\varepsilon^1 u^\varepsilon)_{i+\frac{1}{2},j} - q_{i,j}^1| \leq \sum_{\|(r,s)\| \leq R} \frac{\sum_{m,n=-\nu}^{\nu} |\xi_{m,n}| |\sin(\pi \frac{mr+ns}{2} \varepsilon)|}{1 - \sum_{m,n=-\nu}^{\nu} |\xi_{m,n}| |\sin(\pi \frac{mr+ns}{2} \varepsilon)|} \frac{|\sin \frac{\pi r\varepsilon}{2}|}{\varepsilon} |\hat{u}(r,s)| \leq C\varepsilon$$

when  $\varepsilon > 0$  is small enough. In the same way, one builds  $q_{i,j}^2$  which satisfies (16) and such that

$$|(D_\varepsilon^2 u^\varepsilon)_{i,j+\frac{1}{2}} - q_{i,j}^2| \leq C\varepsilon.$$

It follows that (15) holds with an error of order  $O(\varepsilon)$ .

Observe to conclude that this construction is compatible with the Neumann boundary conditions described in Section 2.3. One can consider (for each  $N$ ) only the values of  $q_{i,j}$  for  $i, j = -\nu, \dots, N + \nu$ , the values of  $p$  restricted to  $Y$ , and restrict  $\mathbf{F}, \mathbf{F}^*$  to these values; in this setting the construction builds an admissible recovery sequence.  $\square$

**Compactness** Theorem 2.4 remains incomplete without the following associated compactness result, whose proof is relatively easy and standard.

**Proposition 2.5.** *Under the same assumptions as in Theorem 2.4, let  $(u^\varepsilon)_{\varepsilon>0}$  with:*

$$\sup_{\varepsilon>0} TV_\varepsilon(u^\varepsilon) < +\infty.$$

*Assume moreover  $u^\varepsilon$  remains bounded (on average, or in some  $L^p(\Omega)$ ). Then there is a subsequence  $(u^{\varepsilon_j})$  and  $u \in L^1(\Omega)$  such that  $u^{\varepsilon_j} \rightarrow u$  in  $L^1(\Omega)$ .*

*Proof.* The point is that by assumption  $q^\varepsilon$  is uniformly bounded, as well as the discrete derivative  $\mathbf{D}_\varepsilon u^\varepsilon = \mathbf{F}^* q^\varepsilon$ . It easily follows that  $(u^\varepsilon)_\varepsilon$  have uniformly bounded total variation, and the claim can be deduced.  $\square$

### 3. Learning a better discretization

In the previous section, we have shown consistency of the general form (4) of the discrete total variation, in the spirit of  $\Gamma$ -convergence, for any interpolation (averaging) kernel  $\mathbf{F}$ . The aim of this section is to investigate numerical methods to learn the “best” interpolation operators  $\mathbf{F}$  given a set of input images and corresponding ground truth solutions.

To do so, we will consider several total variation minimization tasks, where explicit solutions are known (e.g. denoising of a disk, image inpainting) but we will also consider the case of natural image denoising. It will turn out that the learned interpolation operators highly depend on the specific imaging task. That is, an interpolation operator that is found to work well for image denoising will not necessarily work well for image inpainting and vice versa. The reason is that there is a complex interaction between the total variation term and the data fidelity term. For example, in image denoising, the total variation should better smooth nicely the solution (whereas the recovery of sharp edges is mostly driven by the data term), while for image inpainting we expect it to discriminate sharp edges from smoother ones to produce sharper discontinuities.

In what follows, we consider the general class of total variation minimization problem:

$$\min_{Du=F^*q} \lambda \|q\|_Z + G(u, g), \quad (18)$$

where  $G(u, g)$  is a convex data fidelity term depending on a given input image  $g$ , which defines the type of imaging application, e.g. denoising, inpainting, segmentation etc. The parameter  $\lambda > 0$  is used to control the tradeoff between regularization and data fidelity, when needed. We also assume that  $G$  has a proximal map  $\text{prox}_{\tau G}(\cdot)$ ,  $\tau > 0$  which is simple to compute (in closed form). We denote by  $u^*$  a minimizer, unique in case  $G$  is strongly convex.

Since (18) is a difficult non-smooth optimization problem, especially in its primal form, we consider the equivalent saddle-point formulation

$$\min_{u, q} \max_p \langle Du - F^*q, p \rangle + \lambda \|q\|_Z + G(u, g), \quad (19)$$

which can be solved by the (block)-preconditioned primal-dual algorithm [6, 21, 7], whose main iterations are given in Algorithm 1.

**Algorithm 1:** Preconditioned primal-dual algorithm for solving (19)

- Initialization:  $u^0 \in X$ ,  $q^0 \in Z$ ,  $p^0 \in Y$ .
- Step sizes: Choose the block-wise step sizes  $\tau_u, \tau_q, \sigma_p$  such that

$$\|\text{diag}(\sigma_p)^{\frac{1}{2}} (D, F^*) \text{diag}(\tau_u, \tau_q)^{\frac{1}{2}}\| \leq 1,$$

and set  $\theta = 1$ .

- Iterations: For  $k = 0, \dots, K - 1$  let

$$\begin{cases} p^{k+1} = p^k + \sigma_p(Du^k - F^*q^k) \\ \bar{p}^{k+1} = p^{k+1} + \theta(p^{k+1} - p^k) \\ u^{k+1} = \text{prox}_{\tau_u G}(u^k - \tau_u D^* \bar{p}^{k+1}) \\ q^{k+1} = \text{shrink}_{\tau_q \lambda}(q^k - \tau_q F \bar{p}^{k+1}) \end{cases} \quad (20)$$

- Output: Approximate saddle point  $(u^K, q^K, p^K)$

The advantage of block-wise step sizes is that one can easily adapt to different scalings in the linear operators  $\mathbf{D}$  and  $\mathbf{F}^*$ . The proximal map  $\text{prox}_{\tau_u G}$  is assumed to be commutable in closed form, see [6] for several examples which will also be used here. The function  $\text{shrink}_\tau$  denotes the classical  $\ell_{2,1}$  norm shrinkage function which is given for each of the pairs  $\mathbf{q}^l = (q^{l,1}, q^{l,2})$  in  $\mathbf{q} = (\mathbf{q}^1, \dots, \mathbf{q}^L)$  by

$$\hat{\mathbf{q}}^l = \text{shrink}_\tau(\bar{\mathbf{q}}^l) \iff \hat{q}_{i,j}^l = \left(1 - \frac{1}{\max\{1, |\bar{q}_{i,j}^l|/\tau\}}\right) \bar{q}_{i,j}^l, \quad l = 1, \dots, L$$

### 3.1. The learning problem

Clearly, we are here in the classical setting of supervised learning. That is we assume we have given a set of input images  $\mathcal{G} = \{g_1, \dots, g_S\}$  and corresponding target images (explicit solutions)  $\mathcal{T} = (t_1, \dots, t_S)$ . The learning problem is now to find interpolation kernels  $\mathbf{F}$  such that solutions  $u_s^*$  of (19) for input images  $g_s$  minimize a certain distance measure to the target images  $t_s$ .

In order to measure the distance (or loss) between  $u_s^*$  and  $t_s$  we consider a convex and continuously differentiable loss function  $\ell(u, t)$ . Here, we adopt here the classical quadratic loss function  $\ell(u, t) = \frac{1}{2}\|u - t\|^2$ , but any other suitable loss function could be considered as well. The overall loss function with respect to the entire training data set is then given by

$$\mathcal{L}(\mathbf{F}) = \frac{1}{MNS} \sum_{s=1}^S \ell(u_s^*(\mathbf{F}), t_s),$$

where we have made explicit the dependence of the solutions  $u_s^*(\mathbf{F})$  on the interpolation kernels  $\mathbf{F}$ . We also include a pre-factor  $\frac{1}{MNS}$  in order to normalize the loss function with respect to the total number of image pixels.

The learning problem is now given by the following bilevel optimization problem [18]

$$\begin{aligned} & \min_{\mathbf{F}} \mathcal{L}(\mathbf{F}) + \mathcal{R}(\mathbf{F}), \\ & u_s^* \in \arg \min_{u, \mathbf{q}} \max_{\mathbf{p}} \langle \mathbf{D}u - \mathbf{F}^* \mathbf{q}, \mathbf{p} \rangle + \lambda \|\mathbf{q}\|_Z + G(u, g_s), \quad s = 1, \dots, S \end{aligned} \tag{21}$$

Intuitively, this bilevel optimization problem accounts for finding interpolation kernels  $\mathbf{F}$  such that the solutions of the saddle-point problems (here we are only interested in the primal solution  $u_s^*$ ) minimize the loss function on the entire training set.

The function  $\mathcal{R}(\mathbf{F})$  is an additional regularization functional that can be used to impose certain constraints on  $\mathbf{F}$ . As a general requirement for valid interpolation kernels, we must enforce that the filter coefficients in each individual filter sum up to one, that is  $F^{l,1}, F^{l,2} \in C_{\Sigma=1}$ ,  $l = 1, \dots, L$ , where  $C_{\Sigma=1}$  was defined in (10). Hence we set

$$\mathcal{R}(\mathbf{F}) = \delta_{(C_{\Sigma=1})^{L,2}}(\mathbf{F}) = \sum_{l=1}^L \delta_{C_{\Sigma=1}}(F^{l,1}) + \delta_{C_{\Sigma=1}}(F^{l,2}),$$

here  $\delta_C(x) = 0$  if  $x \in C$ , and  $\delta_C(x) = \infty$  else, is the “indicator function” of a set  $C$ .



During learning we will need to compute the orthogonal projection

$$\hat{\mathbf{F}} = \text{proj}_{(C_{\Sigma=1})^{L,2}}(\bar{\mathbf{F}})$$

which because of the independence of the constraints can be done for each filter separately onto the set  $C_{\Sigma=1}$ . The projection of an arbitrary vector  $\bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_n) \in \mathbb{R}^n$  onto the set  $C_{\Sigma=1}$  is given by

$$\hat{\mathbf{x}} = \text{proj}_{C_{\Sigma=1}}(\bar{\mathbf{x}}) \iff \hat{x}_i = \bar{x}_i + \frac{1 - \sum_{i=1}^n \bar{x}_i}{n}.$$

One can also envisage additional constraints in order to make the learned discretization invariant to certain discrete symmetries. We will come back to such constraints when they are needed in the applications. Note that in our framework all constraints are affine or linear, so that, in fact, the optimization can be considered as unconstrained in a proper linear subspace and the reprojection step is equivalent to considering as a descent direction the gradient of the objective restricted to the subspace.

The learning problem causes several difficulties. First, it is highly non-convex since the higher-level problem (loss function) depends on the solution of a non-smooth saddle-point problem. Second, computing gradients of the higher-level problem with respect to  $\mathbf{F}$  is challenging since it involves the sensitivity of the saddle-point of the lower-level problem with respect to the interpolation kernel  $\mathbf{F}$ .

### 3.2. Computing derivatives

The most straightforward approach to compute the derivatives would be to unravel a certain number of iterations of the primal-dual algorithm (Algorithm 1) and to make use of the machinery of automatic differentiation and backpropagation. However, for hard problems such as inpainting, where one would need to unravel several thousands of iterations one easily runs into problems because of memory limitations. In our setting and implementation of the learning, unraveling only 100 iterations takes already more than 5 GB of main memory.

In order to allow learning for large problems or a large number of iterations, we rely on standard sensitivity analysis to derive in Appendix A a method to compute derivatives of a loss function with respect to the linear operator appearing in the saddle-point problem. Although the theory is valid for smooth and strongly convex objectives, we use in practice the same strategy to compute our gradients. In order to simplify the presentation in this section, we only show the computation of the gradient of the loss function with respect to one datum  $(g_s, t_s)$  and we also drop the index  $s$  to simplify the notation. The gradient with respect to the entire data set can be easily computed by just summing up the gradients from the individual data examples.

Assume we have computed a saddle-point  $u^*, \mathbf{q}^*, \mathbf{p}^*$  of (19) for a certain training example  $(g, t)$ . Then, according to (31), the next step is to solve for the adjoint state  $U^*, \mathbf{Q}^*, \mathbf{P}^*$  which is also the solution of the bi-quadratic saddle point problem

$$\min_{U, \mathbf{Q}} \max_{\mathbf{P}} \langle D U - \mathbf{F}^* \mathbf{Q}, \mathbf{P} \rangle + \frac{1}{2} \langle \nabla^2 G(u^*) U, U \rangle - \frac{1}{2} \langle \nabla^2 \|\mathbf{q}^*\|_Z \mathbf{Q}, \mathbf{Q} \rangle + \langle \nabla \ell(u^*, t), U \rangle. \quad (22)$$

Instead of doing this sequentially, we propose a “Piggyback”-style algorithm [15] that jointly computes  $u^*, \mathbf{q}^*, \mathbf{p}^*$  and its adjoint state  $U^*, \mathbf{Q}^*, \mathbf{P}^*$ . Algorithm 2 gives the main iterations of the Piggyback primal-dual algorithm. It is the adaption to our setting of the Algorithm described Section A.2, whose convergence is proved in a smoother setting.

**Algorithm 2:** Piggyback primal-dual algorithm for solving (19) and its adjoint.

- Initialization:  $u^0, U^0 \in X, \mathbf{q}^0, \mathbf{Q}^0 \in Z, \mathbf{p}^0, \mathbf{P}^0 \in Y$ .
- Step sizes: Choose the block-wise step sizes  $\tau_u, \tau_q, \sigma_p$  such that

$$\|\text{diag}(\sigma_p)^{\frac{1}{2}} (\mathbf{D}, \mathbf{F}^*) \text{diag}(\tau_u, \tau_q)^{\frac{1}{2}}\| \leq 1,$$

and set  $\theta = 1$ .

- Iterations: For each  $k = 0, \dots, K - 1$  let

$$\begin{cases} \mathbf{p}^{k+1} = \mathbf{p}^k + \sigma_p(\mathbf{D}u^k - \mathbf{F}^*\mathbf{q}^k), & \mathbf{P}^{k+1} = \mathbf{P}^k + \sigma_p(\mathbf{D}U^k - \mathbf{F}^*\mathbf{Q}^k) \\ \bar{\mathbf{p}}^{k+1} = \mathbf{p}^{k+1} + \theta(\mathbf{p}^{k+1} - \mathbf{p}^k), & \bar{\mathbf{P}}^{k+1} = \mathbf{P}^{k+1} + \theta(\mathbf{P}^{k+1} - \mathbf{P}^k) \\ \tilde{u}^{k+1} = u^k - \tau_u \mathbf{D}^* \bar{\mathbf{p}}^{k+1}, & \tilde{U}^{k+1} = U^k - \tau_u(\mathbf{D}^* \bar{\mathbf{P}}^{k+1} + \nabla \ell(u^k, t)) \\ u^{k+1} = \text{prox}_{\tau_u G}(\tilde{u}^{k+1}), & U^{k+1} = \nabla \text{prox}_{\tau_u G}(\tilde{u}^{k+1}) \cdot \tilde{U}^{k+1} \\ \tilde{\mathbf{q}}^{k+1} = \mathbf{q}^k - \tau_q \mathbf{F} \bar{\mathbf{p}}^{k+1}, & \tilde{\mathbf{Q}}^{k+1} = \mathbf{Q}^k - \tau_q \mathbf{F} \bar{\mathbf{P}}^{k+1} \\ \mathbf{q}^{k+1} = \text{shrink}_{\tau_q \lambda}(\tilde{\mathbf{q}}^{k+1}), & \mathbf{Q}^{k+1} = \nabla \text{shrink}_{\tau_q \lambda}(\tilde{\mathbf{q}}^{k+1}) \cdot \tilde{\mathbf{Q}}^{k+1} \end{cases} \quad (23)$$

- Output: Approximate saddle point  $(u^K, \mathbf{q}^K, \mathbf{p}^K)$  and corresponding adjoint state  $(U^K, \mathbf{Q}^K, \mathbf{P}^K)$

Finally, as shown in (33), the gradient of  $\ell(u^*, t)$  with respect to  $\mathbf{F}$  is given by

$$\begin{aligned} \langle \nabla_{\mathbf{F}} \ell(u^K, t), \mathbf{F} \rangle &= -\langle \mathbf{Q}^K, \mathbf{F} \mathbf{p}^K \rangle - \langle \mathbf{q}^K, \mathbf{F} \mathbf{P}^K \rangle \\ &\iff \nabla_{\mathbf{F}} \ell(u^K, t) = -(\mathbf{Q}^K \otimes \mathbf{p}^K + \mathbf{q}^K \otimes \mathbf{P}^K). \end{aligned} \quad (24)$$

As mentioned, the gradient of the loss function on the whole data set is computed as

$$\nabla \mathcal{L}(\mathbf{F}) = \frac{1}{MNS} \sum_{s=1}^S \nabla_{\mathbf{F}} \ell(u_s^K(\mathbf{F}), t_s). \quad (25)$$

*Remark 3.1.* The expression (24) of the gradient in this form is a bit abstract, but can be easily computed by means of automatic differentiation techniques since expressions such as  $\mathbf{F} \mathbf{p}$  or  $\mathbf{F}^* \mathbf{q}$  are implemented as convolutions with particular boundary conditions. In all our experiments we use the software package Pytorch to compute the derivatives which also takes care of the boundary conditions in the convolutions. In a similar fashion, we compute the Jacobian of the proximal maps times vector products  $\nabla \text{prox}_{\tau_u G}(\tilde{u}^{k+1}) \cdot \tilde{U}^{k+1}$  and  $\nabla \text{shrink}_{\tau_q \lambda}(\tilde{\mathbf{q}}^{k+1}) \cdot \tilde{\mathbf{Q}}^{k+1}$  appearing in Algorithm 2 by means of automatic differentiation. Note that the respective proximal maps might not be continuously differentiable. This can be accounted for by either smoothing the proximal

maps or by selecting an arbitrary sub(super)-gradient which is the standard procedure in the deep learning community for non-smooth activation functions such as the widely used rectified linear unit  $\text{ReLU}(t) = \max\{0, t\}$ . We found the latter option to work well in practice and hence did not consider smoothing.

### 3.3. Learning algorithm

Having detailed the computation of the gradient of the loss function with respect to the interpolation kernel  $\mathbf{F}$ , we now consider the actual learning algorithm. Here, we use an inertial proximal gradient method whose main iterations are presented in Algorithm 3.

**Algorithm 3:** Proximal gradient method for solving (21)

- Initialization:  $\mathbf{F}^{-1} = \mathbf{F}^0 \in (C_{\Sigma=1})^{L,2}$ .
- Step sizes: Choose  $\alpha^k > 0$ ,  $\beta^k \in [0, 1)$ .
- Iterations: For  $k = 0, \dots, K - 1$  let

$$\begin{cases} \bar{\mathbf{F}}^k = \mathbf{F}^k + \beta^k (\mathbf{F}^k - \mathbf{F}^{k-1}) \\ \mathbf{F}^{k+1} = \text{proj}_{(C_{\Sigma=1})^{L,2}} (\bar{\mathbf{F}}^k - \alpha^k \nabla \mathcal{L}(\bar{\mathbf{F}}^k)) \end{cases} \quad (26)$$

- Output: Learned interpolation kernels  $\mathbf{F}^K$

Depending on the particular learning problem we will make use of different settings of the step size  $\alpha^k$  and the inertial parameter  $\beta^k$ .

## 4. Numerical results

In this section we make use of the algorithms of the previous section and show how one can learn the “best” interpolation kernels for different imaging applications. We start with the most simple problem which is the recovery (inpainting) of straight discontinuities but will also consider the regularization (smoothing) of the characteristic functions of disks and natural image denoising.

**Data generation** In the rest of this section we assume a domain  $\Omega = [-1, 1]^2$  which is discretized into  $N \times N$  square pixels of size  $h \times h$ , hence  $h = 2/N$ . According to (5), the values  $u_{i,j}$ , with  $1 \leq i, j \leq N$  of the discrete images correspond to pixel averages of the continuous functions  $u(x)$ .

In order to generate the discrete input images and corresponding ground truth solutions, we precompute pseudo-continuous images by first point-sampling the continuous functions on a regular high-resolution grid of  $BN \times BN$  points and then generate the discrete input images  $g_s$  and corresponding target images  $t_s$ ,  $s = 1, \dots$ , at the target resolution  $N \times N$  via block averaging with blocks of size  $B \times B$ . In all experiments we used a block size of  $B = 64$ .

We also include in all generated synthetic data random shifts of the grid of the size  $h/2$  in order to simulate partial volume effects which is present in any real-world application. This sort of “data augmentation” is a well known technique in deep learning in order to make the learning more robust and generalize better to new data. In fact in our experiments it turned out that it is a simple way to remove inherent data symmetries which could lead to persistent local minima.

As it is standard in machine learning, we always make use of a training set to learn the filters and then evaluate the learned filters on test data which was not part of the training set. For the synthetic images, it will usually mean that the training set differs from the test set by the random shifts.

**Filter sizes** In all synthetic experiments, where the structures to be reconstructed are basically lines or circles, we consider only a small neighborhood of  $2 \times 2$  pixels, which corresponds to convolution kernels of size  $2 \times 3$  for the horizontal component and  $3 \times 2$  for the vertical component of the dual variable  $\mathbf{p}$ . See also Figure 1 and note that the duals are “living” on the faces of the pixels. For natural image denoising, where locally the image structures are much more complex, we will also consider learning with larger neighborhoods.

**Number of filters** We experimented with different numbers  $L$  of filter pairs. In order to investigate the influence of  $L$  to the reconstruction quality we performed experiments using  $L \in \{2, 3, 4, 8\}$  filter pairs. For natural images, we will also perform experiments with a larger number of filter pairs.

**Filter symmetries** Depending on the number of filters it is natural to consider different symmetries which should ideally also reflect the symmetries of the data. For example images are known to be translation invariant, which is naturally captured by the convolution operation but images are also largely rotational invariant, which can be enforced by imposing a rotational invariance on the filters to be learned. Enforcing symmetries also reduces the effective parameters to be learned and one can expect that such learned filters generalize better to new data.

In the case of  $L = 2$  or  $L = 3$  we cannot impose a full rotational invariance and hence it is most natural to consider “only” a transpose symmetry as this symmetry group has in fact two elements. Interestingly, it will turn out that such a transpose symmetry is automatically learned (approximately) without enforcing it explicitly.

For  $L = 4$  and  $L = 8$  it is more natural to consider a rotational symmetry by  $4 \times \frac{\pi}{2}$ , as this symmetry group has four elements.

In order to project the filters onto those symmetry groups, we first perform the projections onto the set  $C_{\Sigma=1}$  followed by a projection onto the respective symmetry group. As an example, for the transpose symmetry for  $L = 2$ , we used the following constraints:

$$F^{1,1} = (F^{2,2})^\top, \quad F^{1,2} = (F^{2,1})^\top,$$

onto which the projection is done by averaging the corresponding filter entries. Overall, this can be shown to give the orthogonal projection onto the intersection of both spaces.

**Filter initialization** As the learning problem represents a nonconvex optimization problem, the initialization has an important impact on the outcome. One can consider different random initializations (uniform, Gaussian, ...) quite common in the deep learning literature, but for our problem it turns out to be better to initialize the filters weights by the linear interpolation weights obtained from interpolating the dual variables on certain regular or random positions. Figure 3 below shows the initial filters in case  $L = 8$ . Note that such initial filters correspond to the choice of the filter weights in the CD discretization.

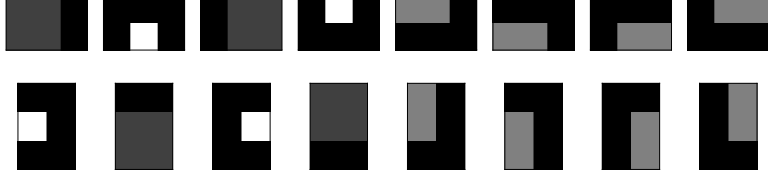


Figure 3: Initialization for  $L = 8$  filter pairs using the weights of linear interpolation, where white corresponds to a weight of 1.0, gray corresponds to a weight of 0.5 and dark gray corresponds to a weight of 0.25.

**Implementation and hardware** All algorithms are implemented in Python with the support of GPU processing and automatic differentiation from the Pytorch package. All code will be made available by the authors. The experiments were performed on an Nvidia Titan RTX eGPU equipped with 24GB memory connected to a Dell XPS Laptop with a Quad-Core i9 CPU, 32GB of main memory, and running Ubuntu 20.04.

#### 4.1. Inpainting

We first consider the problem of inpainting straight discontinuities of various orientations from a given boundary datum. As seen on Fig. 2, most handcrafted discretizations lead to more or less suboptimal solutions and hence the aim of this section is to learn the best set of interpolation kernels which ensure a sharp discontinuity over a large range of orientations. The general problem (18) can be specialized to inpainting by choosing

$$G(u, g) = \sum_{(i,j) \in \mathcal{I}} \delta_{\{g_{i,j}\}}(u_{i,j}),$$

where  $\mathcal{I}$  is a given set which contains the pixel indices corresponding to the given boundary datum and  $\delta_C(\cdot)$  is the indicator function of the convex set  $C$ . We also denote the inpainting domain  $\bar{\mathcal{I}} = \{1, \dots, N\}^2 \setminus \mathcal{I}$ . The proximal map of  $\tau G$  is given by

$$\hat{u} = \text{prox}_{\tau G}(\bar{u}, g) \Leftrightarrow \hat{u}_{i,j} = \begin{cases} g_{i,j} & \text{if } (i, j) \in \mathcal{I} \\ \bar{u}_{i,j} & \text{if } (i, j) \in \bar{\mathcal{I}}. \end{cases}$$

**Training and test data** The training data consists of  $S = 64$  images, each of size  $N \times N$  pixels, with  $N = 64$  and uniformly sampled orientations  $\theta_s = 2s\pi/S$ ,  $s = 0, \dots, S - 1$  of

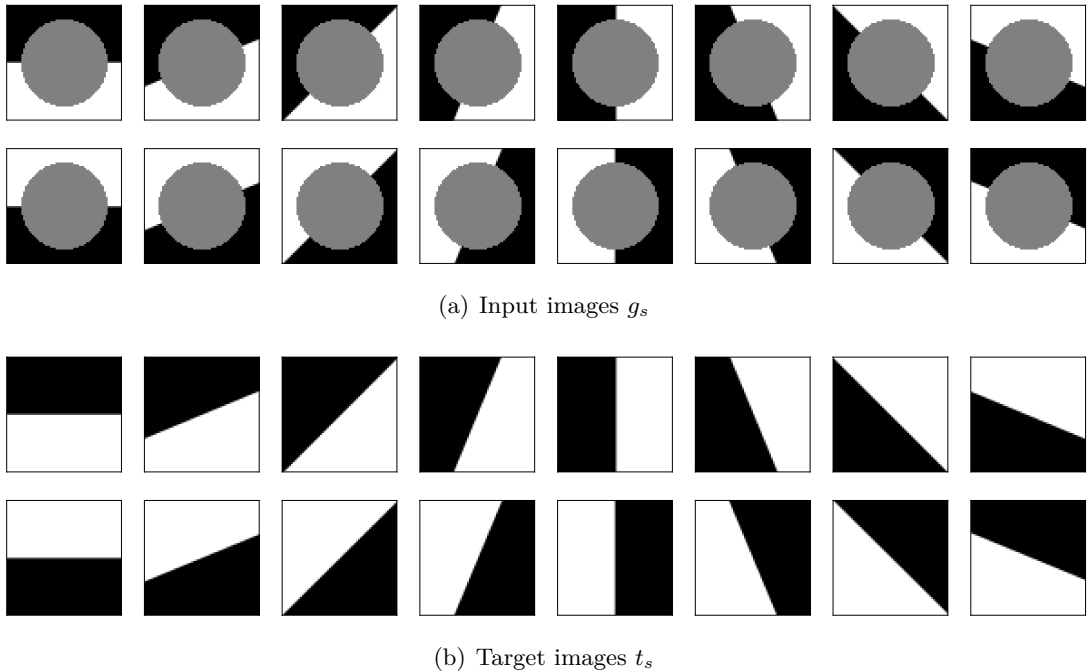


Figure 4: Samples of the training data consisting of the input images  $g_s$  and corresponding target images  $t_s$  for learning to inpaint a straight discontinuity. The disk-like inpainting masks  $\bar{\mathcal{I}}$  are indicated by the gray areas.

the discontinuities. Figure 4 shows a few examples of the training data consisting of the input images  $g_s$  and the corresponding target images  $t_s$ . The inpainting domains  $\bar{\mathcal{I}}$  are shown in gray. As already mentioned above, we also include a small random shift of size  $h/2$  of the discontinuity from the center of the image in order to simulate partial volume effects. The test data was generated in exactly the same way.

**Setting of the learning algorithm** First of all, solving the inpainting problem itself is already a very hard convex optimization problem (in fact similar problems are considered to derive worst-cases rates in convex optimization) and hence one typically needs thousands of iterations. Therefore we must expect that learning is even harder!

The gradients of the lower-level problem were computed by performing  $K = 2000$  iterations of the Piggyback primal-dual algorithm (Algorithm 2). We also used a warm-starting strategy for both the variables and their adjoint states in order to improve the accuracy of the gradients. Note that due to memory limitations, it would be impossible to unravel such a high number of iterations of the primal-dual algorithm and compute the gradient via standard backpropagation.

In the learning algorithm (Algorithm 3) we use a diminishing step size rule of the form  $\alpha^k = \frac{\alpha^0}{\sqrt{k+1}}$  for some  $\alpha^0 > 0$ , as this choice is known to give an optimal convergence rate for the iterates in subgradient methods. The inertial parameter was set to  $\beta^k = 0$  as for this problem we have to expect error in the gradient computation and inertial methods are known to accumulate errors of the gradient.

We stopped the learning algorithm when the loss function did not show any significant improvement, which was usually the case after  $K = 1000 - 2000$  iterations.

**Results** Table 2 plots the learned filters for different settings of the parameter  $L$ . First, all learned filters appear significantly different from the known handcrafted ones. Second, the filters  $L = 2$  and  $L = 3$  (where we did not enforce transpose symmetry) show an almost transpose symmetry, very similar to the transpose symmetric filters  $L = 2(s)$  and  $L = 3(s)$ . This shows that it is absolutely reasonable to enforce symmetry constraints. Moreover, it reduces the effective number of learnable parameters, indeed in case of  $L = 4(s)$  we effectively learned only 2 filters. Third, it is interesting that the learned filters also contain negative values. For example the filter coefficients of the first filter pair  $\mathbf{F}^1 = (F^{1,1}, F^{1,2})$  for the setting  $L = 2(s)$  are given by (rounded to three decimals):

$$F^{1,1} = \begin{pmatrix} 0.002 & 0.205 & 0.334 \\ -0.021 & 0.161 & 0.319 \end{pmatrix}, \quad F^{1,2} = \begin{pmatrix} -0.029 & -0.033 \\ -0.095 & 1.301 \\ -0.007 & -0.137 \end{pmatrix}$$

We evaluated the learned filters on the test data, which consists of the same covering of orientations but with different partial volume effects. We used a number of  $K = 10^5$  iterations of the primal-dual algorithm (Algorithm 1) to compute the solutions. Such a high number of iterations is absolutely necessary to obtain high-accuracy solutions, as the convergence of the primal-dual algorithm is very slow (in fact  $O(1/K)$ ) for the inpainting problem.

Table 2: Learned filters for the inpainting problem where symmetric filters are marked by  $(s)$ .

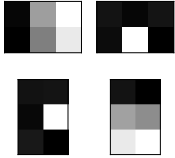
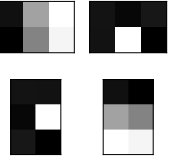
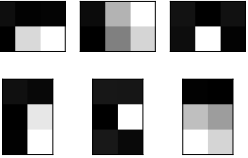
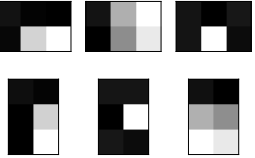
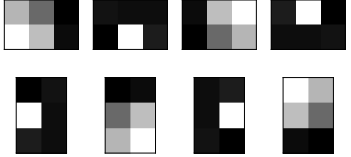
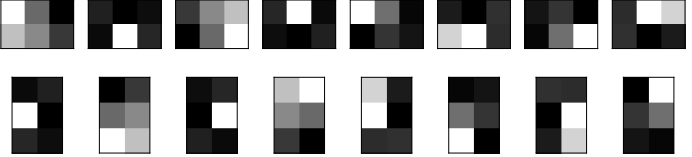
$L = 2$	$L = 2(s)$	$L = 3$	$L = 3(s)$
			
$L = 4(s)$		$L = 8(s)$	
			

Table 2 summarizes and compares the (half) mean squared error ( $\text{MSE} = \mathcal{L}(\mathbf{F})$ )<sup>5</sup> of state-of-the-art handcrafted discretizations and the learned discretizations. For a better readability, we multiply all MSE values by a factor of  $10^5$ .

<sup>5</sup>Please not the additional factor of  $\frac{1}{2}$  in  $\mathcal{L}(\mathbf{F})$  which we keep in order to make the MSE compatible our loss function.

First of all, one can see that the MSE is very similar on both the training data and the test data which indicates that overfitting is minor, particularly in cases where we explicitly enforced a certain symmetry constraint. As already discussed, the best handcrafted filters is given by CD, which outperforms the other handcrafted variants by a large margin. One can also see that having a larger number of filters leads to consistently better results, but it is more the symmetry of the filters which is important than the effective number of parameters.

Table 3:  $10^5 \times$  the mean squared error (MSE) of handcrafted and learned filters evaluated on both the training and test data.

Data	FD	RT	CD	$L = 2$	$L = 2(s)$	$L = 3$	$L = 3(s)$	$L = 4(s)$	$L = 8(s)$
Train	135	195	6.69	1.26	1.22	1.19	1.27	0.85	0.77
Test	134	194	6.33	1.63	1.45	1.29	1.29	0.87	0.82

Figure 5 plots the peak signal to noise ratio ( $\text{PSNR} = -10 \log_{10} 2\text{MSE}$ ) of the single test images over the range of orientations. Note that a PSNR larger than 50 can already be considered as almost perfect. From the plot one can see that the learned filters are consistently better compared to the handcrafted filters and are much more rotationally invariant. Note that some handcrafted filters are very good in recovering grid-aligned orientations but at the cost of poor results for non-grid-aligned orientations.

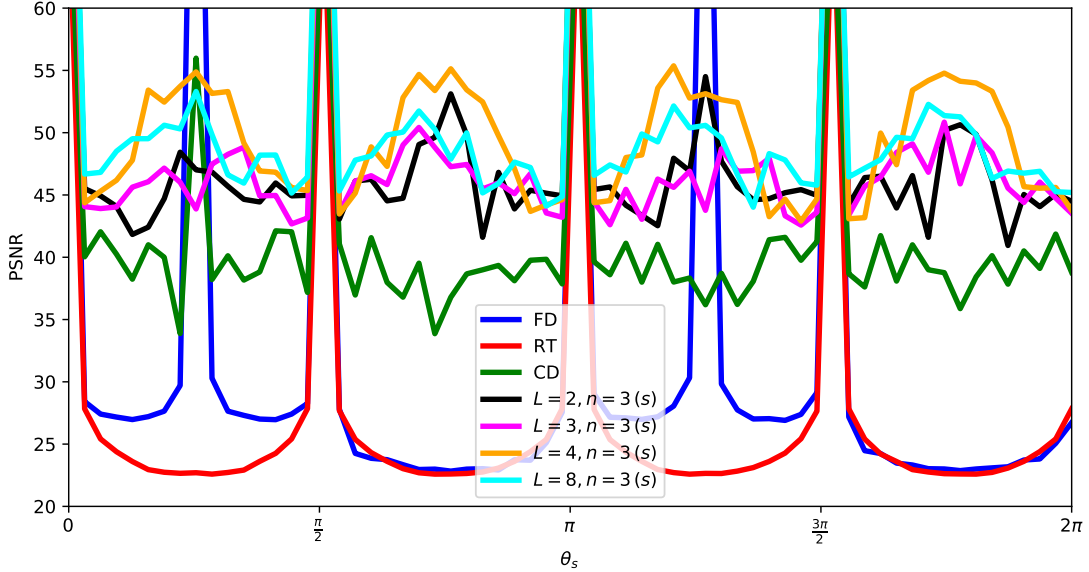


Figure 5: Reconstruction quality (in PSNR) plotted over the angles of the discontinuities. One can see that the learned filters offer a significantly improved isotropy.

Finally, Figure 6 provides false-color error plots of one training example with respect to the corresponding target image. One can see that FD and RT largely fail but CD gives very good results. The learned filters, in particular the setting  $L = 4(s)$ , give still better results (although hard to see with the eye).



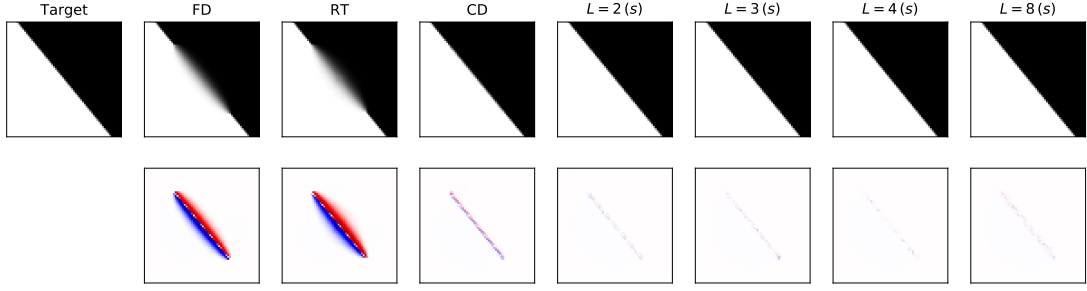
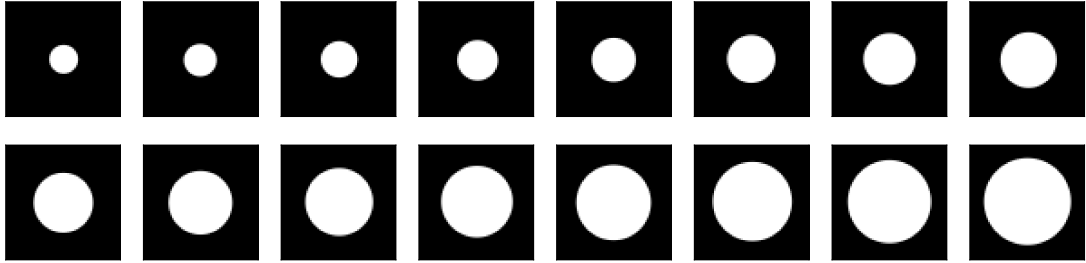
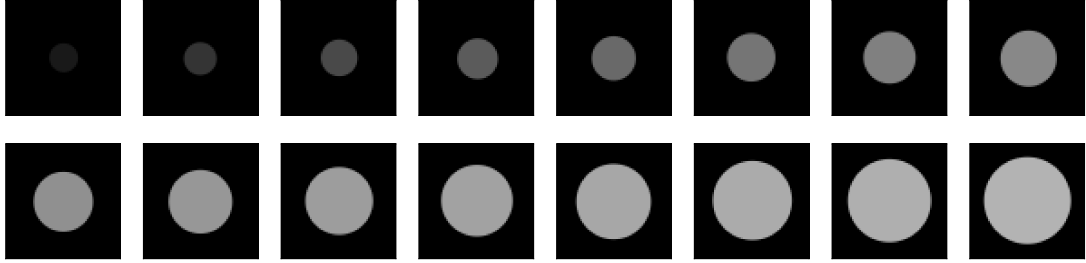


Figure 6: Color coded differences to the ground truth image  $t_s$  for  $s = 56$ ,  $\theta_s \approx \frac{7\pi}{4}$ . CD gives the best result among the hand crafted filters but overall, the learned filters using  $L = 4(s)$  gives the best results.

## 4.2. Disk denoising



(a) Input images  $g_s$



(b) Target images  $t_s$

Figure 7: Samples of the training data for learning to denoise disks of different sizes.

In our second experiment, we consider regularizing (denoising) a white disk over a black background using the ROF model and 0-Dirichlet boundary conditions. The ROF model, is obtained from (18) by setting  $G(u, g) = \frac{1}{2}\|u - g\|_2^2$ . The corresponding proximal map, which is needed in the primal dual algorithm is given by

$$\hat{u} = \text{prox}_{\tau G}(\bar{u}, g) \iff \hat{u}_{i,j} = \frac{\bar{u}_{i,j} + \tau g_{i,j}}{1 + \tau}.$$

It is a well known fact that for the disk denoising problem, the (unique) solution of the

ROF model, is given by a disk of the same radius but with a reduced intensity value  $1 - \frac{2\lambda}{r_s}$ , depending on the radius  $r_s$  and the setting of the regularization parameter  $\lambda$ .

**Training and test data** The training data comprises  $S = 64$  input images  $g_s$  of the characteristic functions of disks with radius  $r_s$ , where we sample  $r_s$  regularly in the range  $[0.25, 0.75]$ . The corresponding target images  $t_s$  are obtained by reducing the intensity value of the disk to  $1 - \frac{2\lambda}{r_s}$ . We used a setting of  $\lambda$  that ensures the existence of a solution for all values of  $r_s$ . Additionally we incorporated a small random shift of the center of the disk of size  $h/2$  to increase the variability of the data. Figure 7 exemplifies a few instances of the training data consisting of the input images and corresponding ground truth solutions. The test data was generated in exactly the same way but with different random shifts.

**Setting of the learning algorithm** Due to the strong convexity of the ROF model (in the variable  $u$ ), the computation of a high-accuracy solution is much easier than for the inpainting problem. Also the learning appears significantly easier in practice. For computing the derivatives, we used  $K = 200$  iterations of the Piggyback algorithm (Algorithm 3), again with a warm-starting strategy for both the variables and their adjoint states. In the learning algorithm we used a constant step size  $\alpha^k = \alpha^0$  with  $\alpha^0 > 0$ . The inertial parameter  $\beta^k$  was set to  $\beta^k = \frac{1}{\sqrt{2}}$ , as this setting is known to lead to a convergent algorithm for optimization problems with  $\frac{1}{\alpha^0}$ -Lipschitz-continuous gradients [20]. The learning was stopped when no significant improvement of the objective function was observed, in practice after approximately  $K = 1000$  iterations.

**Results** Table 4 summarizes the learned filters for the disk denoising problem. At first glance, the filters obtained for disk denoising look very different from the filters learned for inpainting (c.f. Table 2). Indeed the coefficients of the first filter pair for the setting  $L = 2(s)$  obtained are given by (rounded to 3 decimals)

$$F^{1,1} = \begin{pmatrix} -0.001 & 0.294 & 0.349 \\ -0.037 & 0.168 & 0.227 \end{pmatrix}, \quad F^{1,2} = \begin{pmatrix} 0.009 & -0.003 \\ -0.169 & 1.250 \\ -0.026 & -0.062 \end{pmatrix}$$

Moreover, it is interesting that the filters obtained for the setting  $L = 4(s)$  are very similar (up to a translation) to the handcrafted filters of RT. This is also confirmed by the MSE values in Table 5 where RT appears to be a very well performing handcrafted filter. From the table one can also see that learning consistently improves on the entire data set (training set and testing set). The best learned variant is given by the setting  $L = 8(s)$ . Figure 8 additionally plots for the testing data the PSNR values for the different handcrafted and learned filters over the radius of the disk. One can observe that the learned filters consistently perform better compared to the handcrafted filters over the whole range of radii. The simple (de facto standard) discretization FD gives worst results, both in terms of the sharpness of the boundary of the disk and the value of the disk itself. Indeed, one weakness of all handcrafted filters seem to be to identify correctly the value of the disk which results in a large error. This is exactly, where

learning improves and finds a discretization which gives a better value of the disk. See Figure 8 for a color coding based visualization of this issue.

Table 4: Learned filters for the disk denoising problem. Symmetric filters are indicated by  $(s)$ .













































$L = 2$		$L = 2(s)$		$L = 3$			$L = 3(s)$				
											
											
$L = 4(s)$				$L = 8(s)$							
											
											

Table 5:  $10^5$  times the mean squared error (MSE) of handcrafted and learned filters for the disk denoising problem.

Data	FD	RT	CD	$L = 2$	$L = 2(s)$	$L = 3$	$L = 3(s)$	$L = 4(s)$	$L = 8(s)$
Train	22.28	1.36	2.33	2.10	2.10	1.62	1.63	0.73	0.48
Test	22.36	1.32	2.30	2.10	2.10	1.60	1.60	0.72	0.47

### 4.3. Natural image denoising

We now propose to learn the “best” filters for denoising natural images. Of course we have to admit that we do not expect the total variation to be competitive against state-of-the-art image denoising methods, it is just interesting to see which kind of filters are learned for natural image denoising and how they compare to the filters learned on the synthetic images. We will consider the same setting as in the previous experiments, but will also present results of a larger experiment where we learn 40 filter pairs with an effective neighborhood of  $6 \times 6$  pixels. Despite the fact we are restricted to the setting of consistent discretizations of the total variation, at the lowest scale this introduces a flavor of non-locality which is known to be the key to the success of state-of-the-art approaches. We also present results in this setting enforcing an additional  $90^\circ$  rotational invariance, so that the effective number of filter pairs is reduced to 10.

For natural image denoising, we still consider the ROF model, now with Neumann boundary conditions. Since we are dealing with natural images, we do not know (even if we know the noise level) the optimal regularization parameter  $\lambda > 0$ . To overcome this issue, we fix in our optimization a value of  $\lambda$  but replace the sum constraint (10) by the

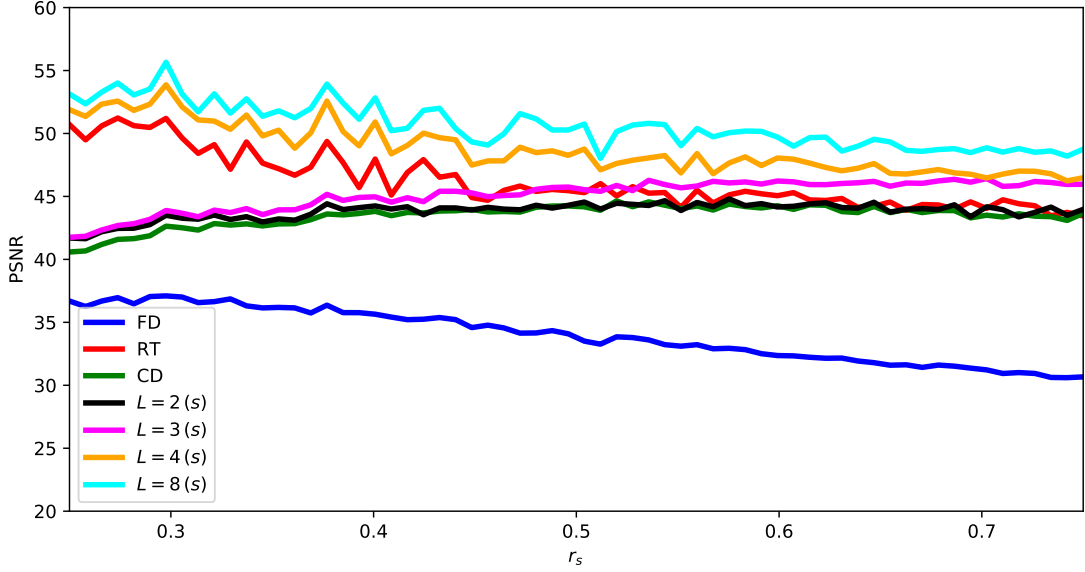


Figure 8: Reconstruction errors measured in PSNR for different handcrafted and learned filters for the disk denoising problem.

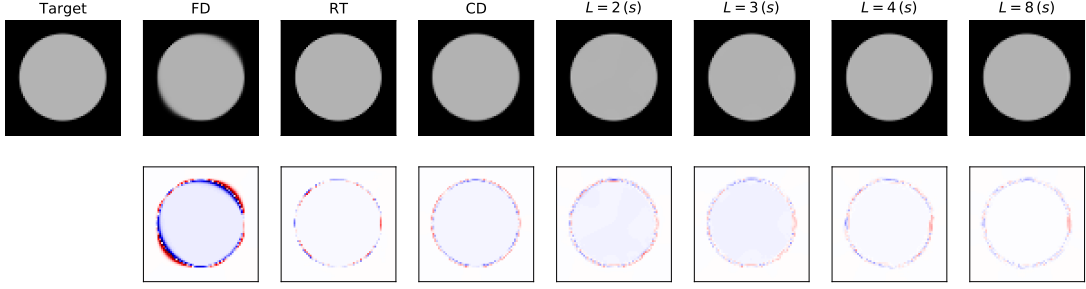


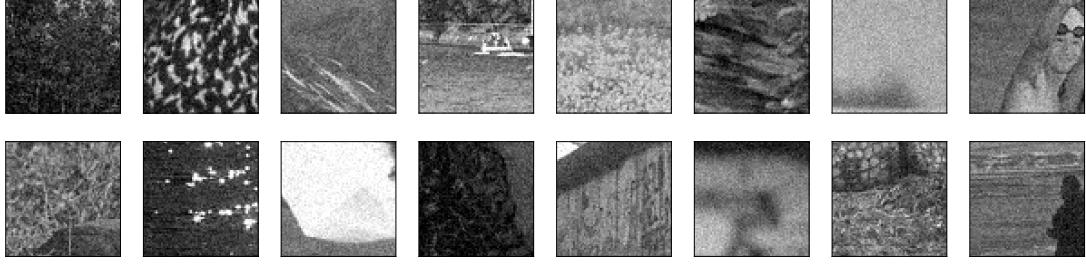
Figure 9: Color coded differences to the ground truth image  $t_s$  with  $s = 64$ . RT gives the best result among the handcrafted filters but the value of the disk is well reconstructed only for the learned filter  $k = 8, s$ .

looser constraints  $\mathbf{F} \in C_{\Sigma=\mu}$  where:

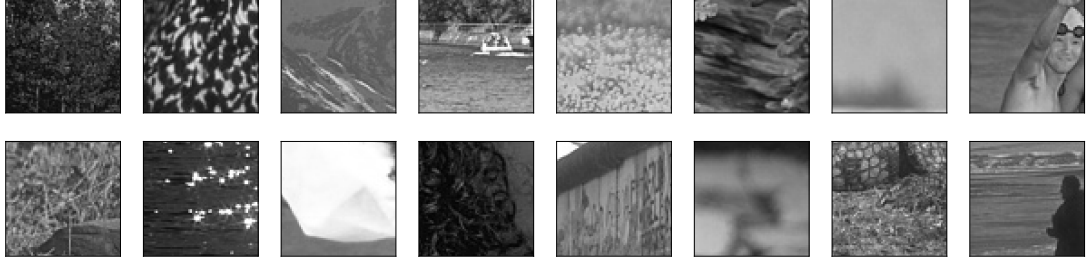
$$C_{\Sigma=\mu} = \{\mathbf{F} : \exists \mu \in \mathbb{R}, \sum_{m,n} \xi_{m,n}^l = \sum_{m,n} \eta_{m,n}^l = \mu, l = 1, \dots, L\}. \quad (27)$$

In words, the sum of the coefficients of all the filters of  $\mathbf{F}$  should be equal to the same value. Exploiting the 1-homogeneity of the norm, the “optimal” value of the regularization parameter in the original problem (18) is then given by  $\lambda|\mu|$ , and the normalized filter coefficients, which satisfy (10), are simply obtained by dividing all the coefficients of  $\mathbf{F}$  by  $\mu$ .

During learning, we project the coefficients onto the constraint (27). This of course has a simple closed form. Given  $m$  vectors  $\bar{\mathbf{x}}^i = (\bar{x}_1^i, \dots, \bar{x}_n^i) \in \mathbb{R}^n$ ,  $i = 1, \dots, m$ , the



(a) Input images  $g_s$



(b) Target images  $t_s$

Figure 10: Samples of the training data for natural image denoising. The input images are given by adding i.i.d zero-mean Gaussian noise with a standard deviation of  $\sigma = 0.05$ .

projection onto (27) reduces to the problem:

$$\min_{\mu, \{\mathbf{x}_i\}_{i=1}^m} \frac{1}{2} \sum_{i=1}^m \|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2, \quad \text{s.t.} \quad \sum_{j=1}^n x_j^i = \mu, \quad i = 1, \dots, m.$$

Assuming that the optimal solution is attained for some  $\hat{\mu} \in \mathbb{R}$ , then the entries of the projected vectors  $\hat{\mathbf{x}}_i = (\hat{x}_1^i, \dots, \hat{x}_n^i)$  are given by

$$\hat{\mathbf{x}} = \hat{x}_j^i = \bar{x}_j^i + \frac{\hat{\mu} - \sum_{j=1}^n \bar{x}_j^i}{n}, \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

Substituting this formula into the objective function, we get that  $\hat{\mu}$  solves:

$$\min_{\mu} \frac{1}{2} \sum_{i=1}^m (\mu - \sum_{j=1}^n \bar{x}_j^i)^2,$$

from which we obtain

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n \bar{x}_j^i.$$

**Training and test data** For the training data, we used  $S = 64$  target patches  $t_s$ ,  $s = 1, \dots, S$ , each of size  $64 \times 64$  pixels, which are randomly cropped from images of the well-known BSDS500 database [2]. The input patches  $g_s$  are generated by adding

i.i.d. zero-mean Gaussian noise with a standard deviation of  $\sigma = 0.05$ . The test data was generated in exactly the same way, but of course contain a different set of randomly cropped patches. For the handcrafted filters the optimal regularization parameter was determined to be  $\lambda = 0.025$  and kept constant for both the training and test set. For the learned filters we used  $\lambda = 0.02$  but the effective regularization parameter was adapted during learning, by the additional freedom to adjust the sum of the filter coefficients.

**Setting of the learning algorithm** We used exactly the same setting of the learning algorithm as in the case of the disk denoising experiment.

**Results** In Table 6 we plot the learned filters for the setting  $L = 8(s)$  using a neighborhood of  $2 \times 2$  pixels,  $L = 40$  and  $L = 40(s)$  using a neighborhood of  $6 \times 6$  pixels. Comparing the setting  $L = 8(s)$  obtained for natural images with the respective filters obtained for disk denoising, we see that the filters for natural images are significantly different. The main reason is that natural photographs do not contain sharp discontinuities because of the band limitation of optical devices. The larger filters show quite nicely that the learned interpolation kernels correspond to oriented edge- and line-like structures and hence better adapt to the structures of natural images. As mentioned before, another interpretation is that the learned filters correspond to a convolutional dictionary for the local image gradient  $Du$ .

Table 6: Learned filters for natural image denoising. Symmetric filters are indicated by  $(s)$ .

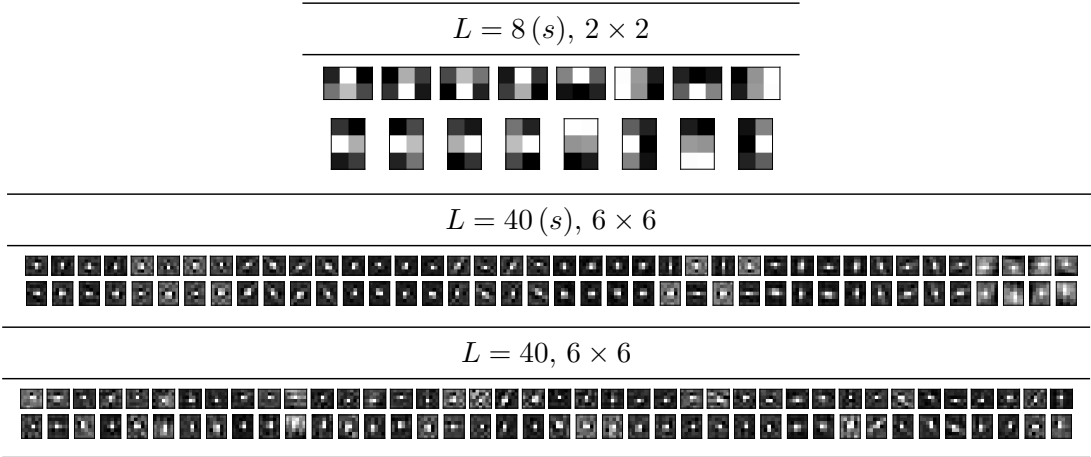


Table 7 presents quantitative results and comparisons with hand-crafted discretizations. One can see that the learned discretizations leads to significantly better values than any of the handcrafted discretizations. Interestingly, while the setting  $L = 40$  yields the smallest training MSE, the setting  $L = 40(s)$ , enforcing the rotational invariance, gives a slightly better test MSE and seems to generalize better to new data.

In Figure 11 we finally provide a per-instance comparison between the PSNR values of the learned discretization  $L = 40(s)$  and the handcrafted discretization CD. One can see that as expected, a clear majority of image instances are better denoised by the

Table 7:  $10^4 \times$  the mean squared error (MSE) of handcrafted and learned filters for natural image denoising.

Data	FD	RT	CD	$L = 8 (s)$	$L = 40 (s)$	$L = 40$
Train	5.05	5.33	4.87	4.58	4.31	4.22
Test	4.72	5.05	4.51	4.28	4.10	4.13

learned discretization. We also exemplify provide one typical denoising results, where the learned discretization yields an improvement of almost 0.8 dB.

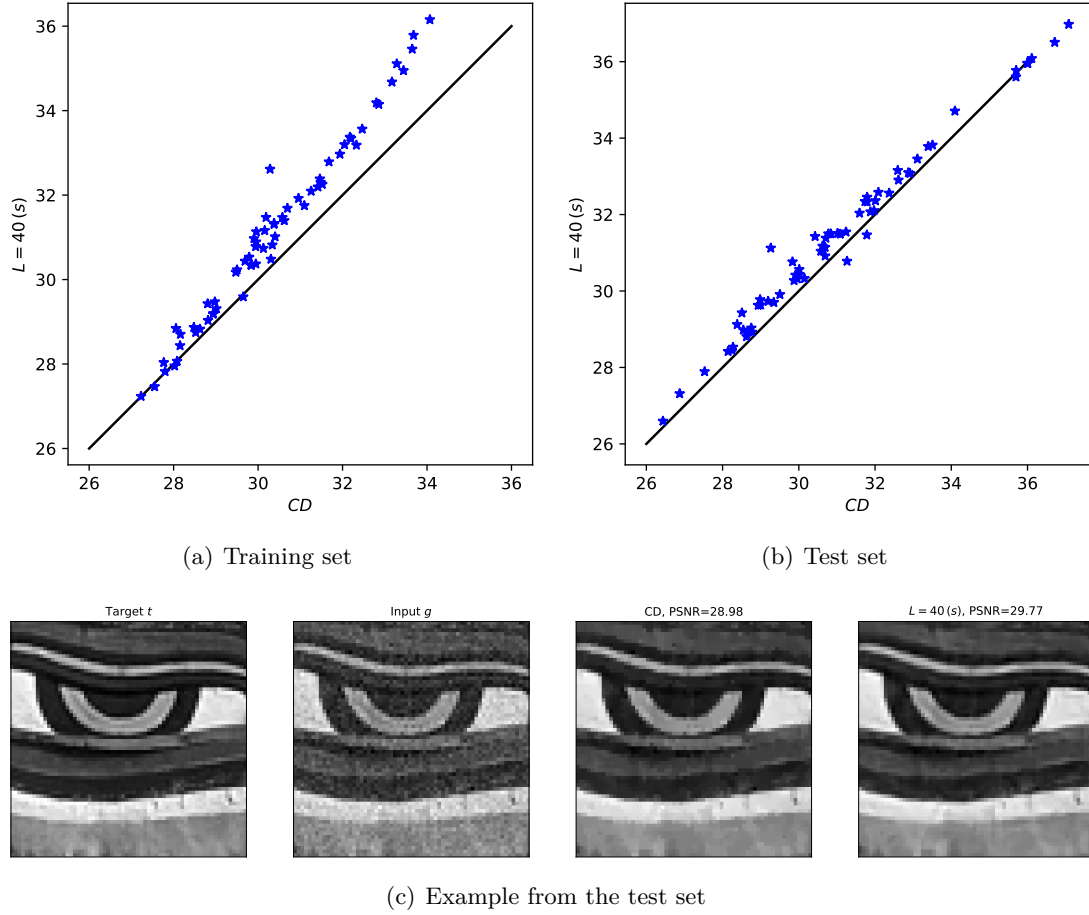


Figure 11: Comparison between the best handcrafted filters (CD) and the learned filters  $L = 40 (s)$  for both the training set (a) and the test set (b). (c) exemplifies a denoised image patch from the test set where the learned discretization of the total variation gives a significant improvement.

Table 8: Confusion matrix for the crossover experiments. For comparison the last column depicts the results for the handcrafted filters of CD.

		Learning task			Handcrafted
		Line	Disk	Natural	CD
Evaluation task	Line	0.82	243.55	50.71	6.33
	Disk	1.88	0.47	4.08	2.30
	Natural	48.68	49.65	42.80	45.10

#### 4.4. Crossover experiments

We conclude this section by carrying out crossover experiments, where we compare the filters learned on one specific task to the other tasks. The aim of such crossover experiments is to investigate the extent to which the learned discretization schemes can be generalized to other tasks. In order to make a relatively fair comparison, we only compare the filter of the setting  $L = 8(s)$ , for which we have obtained good results on all the tasks. In Table 8, we provide a “confusion matrix” of the results and we put the respective results of the handcrafted scheme CD as a reference. One can see that the filters obtained for line inpainting seem to generalize best to the other tasks. Surprisingly, the filters obtained for natural image denoising task seem to generalize better than the filters learned for the synthetic disk denoising task. It seems that for disk denoising the role of the quadratic data fidelity term is too strong such that almost “any” consistent discretization of the total variation may give good results. At this point one could of course try to learn optimal filters for several tasks simultaneously, and one can expect that this gives even better generalization capabilities, but we leave this experiments for future work.

## A. Derivatives of saddle-points

In this appendix, we give a partial justification of the “Piggyback”-style Algorithm 2 the derivative of our loss function. We consider to simplify a smoother objective. This is obtained by assuming that both  $\|\cdot\|_Z$  and  $G(\cdot, g)$  in (19) are replaced with smooth and strongly convex functions: in such situation our sensitivity analysis shows that Algorithm 2 actually converges to states and adjoint states from which one can recover a gradient of the objective. In general, we can just hope that it produces a limit of gradients which would be computed by smoothing the appropriate functions and then letting the smoothing go to zero, but this remains an open question. On the other hand, the good quality of the gradients we compute and the results of the optimization give a hint that it seems to be the case in our application.

### A.1. Theoretical grounds

We consider here a generic problem of type (19), yet in a smoother case. Following the notation in [6, 7], we are given two convex, lower semicontinuous functions  $g, f$  such that  $g, f^*$  are smooth (at least  $C^2$ , we will require a bit more in Sec. A.3) and strongly



convex (defined on some finite dimensional Euclidean spaces  $X, Y$ ). We consider then the standard saddle point problem

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + g(x) - f^*(y) \quad (28)$$

which, with our assumptions, has for any given linear operator  $K : X \rightarrow Y^* \simeq Y$  a unique saddle point  $(\hat{x}, \hat{y})$ . Then, we consider the convex and continuously differentiable loss function

$$\mathcal{L}(K) = \ell(\hat{x}(K), \hat{y}(K)),$$

where we have made explicit the dependence of the saddle point  $(\hat{x}, \hat{y})$  with respect to the linear operator  $K$ . Our aim is to find a derivative of  $\mathcal{L}$  with respect to the linear operator  $K$ .

Following a classical approach in sensitivity analysis, we start from the optimality condition of the saddle-point problem (28):

$$\begin{cases} K\hat{x} - \nabla f^*(\hat{y}) = 0 \\ K^*\hat{y} + \nabla g(\hat{x}) = 0 \end{cases} \quad (29)$$

and assume  $K$  is perturbed by a small variation  $sL$ ,  $|s| \ll 1$ , in a given direction  $L$ . Denoting  $\hat{x}_s = \hat{x} + s\xi_s$  and  $\hat{y}_s = \hat{y} + s\eta_s$  the solution for the linear operator  $K + sL$  it follows that

$$\begin{cases} K\hat{x}_s + s(K\xi_s + L\hat{x}_s) - \nabla f^*(\hat{y}) - (\int_0^s D^2 f^*(\hat{y} + t\eta_s) dt) \eta_s = 0, \\ K^*\hat{y}_s + s(K^*\eta_s + L^*\hat{y}_s) + \nabla g(\hat{x}) + (\int_0^s D^2 g(\hat{x} + t\xi_s) dt) \xi_s = 0. \end{cases}$$

Dividing by  $s$  and making use of the optimality condition (29) yields

$$\begin{cases} K\xi_s + L\hat{x}_s - \left(\frac{1}{s} \int_0^s D^2 f^*(\hat{y} + t\eta_s) dt\right) \eta_s = 0 \\ K^*\eta_s + L^*\hat{y}_s + \left(\frac{1}{s} \int_0^s D^2 g(\hat{x} + t\xi_s) dt\right) \xi_s = 0. \end{cases}$$

Thanks to the strong convexity of  $g, f^*$  (and the continuity of the Hessians) we can pass to the limit as  $s \rightarrow 0$  and see that  $\xi_s, \eta_s$  go to  $\xi, \eta$  satisfying

$$\begin{cases} K^*\eta + L^*\hat{y} + D^2 g(\hat{x})\xi = 0, \\ -K\xi - L\hat{x} + D^2 f^*(\hat{y})\eta = 0, \end{cases}$$

where we have exchanged the order of the equations and changed the sign of the second line. The unique solution is given by

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} D^2 g(\hat{x}) & K^* \\ -K & D^2 f^*(\hat{y}) \end{pmatrix}^{-1} \begin{pmatrix} -L^*\hat{y} \\ L\hat{x} \end{pmatrix}$$

We can now compute the directional derivative  $\mathcal{L}'(K; L) = \langle \nabla \mathcal{L}(K), L \rangle$ , as follows:

$$\mathcal{L}'(K; L) = \nabla \ell(\hat{x}, \hat{y})^T \begin{pmatrix} \xi \\ \eta \end{pmatrix} = \nabla \ell(\hat{x}, \hat{y})^T \begin{pmatrix} D^2 g(\hat{x}) & K^* \\ -K & D^2 f^*(\hat{y}) \end{pmatrix}^{-1} \begin{pmatrix} -L^*\hat{y} \\ L\hat{x} \end{pmatrix}. \quad (30)$$

Next, we introduce adjoint variables  $X, Y$ , such that

$$\begin{aligned} (-X^T, Y^T) &= \nabla \ell(\hat{x}, \hat{y})^T \begin{pmatrix} D^2 g(\hat{x}) & K^* \\ -K & D^2 f^*(\hat{y}) \end{pmatrix}^{-1} \\ \Leftrightarrow \quad \nabla \ell(\hat{x}, \hat{y}) &= \begin{pmatrix} D^2 g(\hat{x}) & -K^* \\ K & D^2 f^*(\hat{y}) \end{pmatrix} \begin{pmatrix} -X \\ Y \end{pmatrix}, \end{aligned}$$

which we can write as:

$$\begin{cases} D^2 g(\hat{x})X + K^*Y + \nabla_x \ell(\hat{x}, \hat{y}) = 0, \\ -KX + D^2 f^*(\hat{y})Y - \nabla_y \ell(\hat{x}, \hat{y}) = 0. \end{cases} \quad (31)$$

Equation (31) are the optimality conditions of the bi-quadratic adjoint saddle-point problem:

$$\min_X \sup_Y \langle KX, Y \rangle + \frac{1}{2} \langle D^2 g(\hat{x})X, X \rangle - \frac{1}{2} \langle D^2 f^*(\hat{y})Y, Y \rangle + \left\langle \nabla \ell(\hat{x}, \hat{y}), \begin{pmatrix} X \\ Y \end{pmatrix} \right\rangle.$$

Denoting by  $(\hat{X}, \hat{Y})$  the unique solution of this adjoint saddle-point problem, the directional derivative (30) is given by

$$\mathcal{L}'(K; L) = \langle \hat{X}, L^* \hat{y} \rangle + \langle \hat{Y}, L \hat{x} \rangle, \quad (32)$$

for any  $L$ . Using  $\mathcal{L}'(K; L) = \langle \nabla \mathcal{L}(K), L \rangle$  we find that the gradient is given by

$$\nabla \mathcal{L}(K) = \hat{Y} \otimes \hat{x} + \hat{y} \otimes \hat{X}. \quad (33)$$

In practice, our variable will be only a subset of the operator  $K$ , in which case it is convenient to assemble the linear form (32) and use automatic differentiation to extract its gradient with respect to the relevant parameters.

## A.2. Algorithms

Algorithm 3 in [6] (or Alg. 5 in [7]) can be implemented to solve both (29) and (31). For the latter it yields the following iterations: one chooses  $\tau, \sigma > 0$ ,  $\theta \leq 1$  (particular choices will be discussed later on), choose  $(X^0, Y^0)$  and let for each  $k$ :

$$\begin{cases} X^{k+1} = (I + \tau D^2 g(\hat{x}))^{-1} (X^k - \tau (K^* Y^k + \nabla_x \ell(\hat{x}, \hat{y}))) \\ \bar{X}^{k+1} = X^{k+1} + \theta (X^{k+1} - X^k) \\ Y^{k+1} = (I + \sigma D^2 f^*(\hat{y}))^{-1} (Y^k + \sigma (K \bar{X}^{k+1} + \nabla_y \ell(\hat{x}, \hat{y}))). \end{cases} \quad (34)$$

Here  $(\hat{x}, \hat{y})$  have to be computed first by solving (29), which can be done using the same algorithm (and actually, with the same parameters  $\tau, \sigma, \theta$ ): we pick  $(x^0, y^0)$  (possibly as the result of a former optimization) and let for each  $k$ :

$$\begin{cases} x^{k+1} = (I + \tau \nabla g)^{-1} (x^k - \tau K^* y^k) \\ \bar{x}^{k+1} = x^{k+1} + \theta (x^{k+1} - x^k) \\ y^{k+1} = (I + \sigma \nabla f^*)^{-1} (y^k + \sigma K \bar{x}^{k+1}). \end{cases} \quad (35)$$

Now, a natural algorithm consists in replacing in (34) the values of  $(\hat{x}, \hat{y})$ , which can be estimated only up to some precision, with the iterate  $(x^k, y^k)$  or  $(x^{k+1}, y^{k+1})$  of the iteration (35). Denoting  $\text{prox}_{\tau g} = (I + \tau \nabla g)^{-1}$ , etc, we observe that for any  $x$ ,

$$\nabla \text{prox}_{\tau g}(x) = (I + \tau D^2 g(\text{prox}_{\tau g}(x)))^{-1}.$$

Hence  $\nabla \text{prox}_{\tau g}(x^k - \tau K^* y^k) = (I + D^2 g(x^{k+1}))^{-1}$ , etc. Computationally, the idea would be to evaluate such a gradient using automatic differentiation, against the vector  $X^k - \tau(K^* Y^k + \nabla_x \ell(\hat{x}, \hat{y}))$ : however also in the latter we have to substitute  $(\hat{x}, \hat{y})$  with known values, which at this point would be  $(x^k, y^k)$ . This would transform (34) into:

$$\begin{cases} X^{k+1} = \nabla \text{prox}_{\tau g}(x^k - \tau K^* y^k) \cdot (X^k - \tau(K^* Y^k + \nabla_x \ell(x^k, y^k))) \\ \bar{X}^{k+1} = X^{k+1} + \theta(X^{k+1} - X^k) \\ Y^{k+1} = \nabla \text{prox}_{\sigma f^*}(y^k + \sigma K \bar{X}^{k+1}) \cdot (Y^k + \sigma(K \bar{X}^{k+1} + \nabla_y \ell(x^k, y^k))). \end{cases} \quad (36)$$

where  $x^k, y^k, \bar{x}^{k+1}$  are computed using (35). Iterations (35) and (36) translate to Alg. 2 in the particular setting of this paper.

### A.3. Convergence analysis

We denote  $\gamma$  the modulus of convexity of  $g$  and  $\delta$  the modulus of  $f^*$ . Thanks to [6, Thm. 3], choosing  $\mu = 2\sqrt{\gamma\delta}/\|K\|$ ,  $1/(1+\mu) \leq \theta \leq 1$ ,  $\tau = \mu/(2\gamma)$  and  $\sigma = \mu/(2\delta)$  we have:

$$\gamma\|x^k - \hat{x}\|^2 + (1 - \omega)\delta\|y^k - \hat{y}\|^2 \leq C\omega^k$$

for  $\omega = (1 + \theta)/(2 + \mu) < 1$  (precisely,  $\omega = \theta$  in case  $\theta = 1/(1 + \mu)$ ,  $\omega = 1/(1 + \mu/2)$  in case  $\theta = 1$ ). Hence, one should expect  $X^k, Y^k$  to converge to the solution of (31) if  $g, f^*$  and  $\ell$  are smooth enough, following the analysis for inexact primal-dual algorithms (see for instance [22]).

We can sketch a convergence analysis, following [6, 7, 22], as follows. We write (34) as

$$\begin{cases} D^2 g(x^{k+1}) \cdot X^{k+1} = \frac{X^k - X^{k+1}}{\tau} - (K^* Y^k + \nabla_x \ell(x^k, y^k)) \\ D^2 f^*(y^{k+1}) \cdot Y^{k+1} = \frac{Y^k - Y^{k+1}}{\sigma} + (K \bar{X}^{k+1} + \nabla_y \ell(x^k, y^k)). \end{cases}$$

Subtracting (31), we get that

$$\begin{cases} D^2 g(x^{k+1}) \cdot (X^{k+1} - \hat{X}) = \frac{X^k - X^{k+1}}{\tau} - K^*(Y^k - \hat{Y}) + e_X^{k+1} \\ D^2 f^*(y^{k+1}) \cdot (Y^{k+1} - \hat{Y}) = \frac{Y^k - Y^{k+1}}{\sigma} + K(\bar{X}^{k+1} - \hat{X}) + e_Y^{k+1} \end{cases}$$

where the error terms are given by

$$\begin{aligned} e_X^{k+1} &= -(\nabla_x \ell(x^k, y^k) - \nabla_x \ell(\hat{x}, \hat{y})) + (D^2 g(x^{k+1}) - D^2 g(\hat{x})) \cdot \hat{X}, \\ e_Y^{k+1} &= \nabla_y \ell(x^k, y^k) - \nabla_y \ell(\hat{x}, \hat{y}) - (D^2 f^*(y^{k+1}) - D^2 f^*(\hat{y})) \cdot \hat{Y}. \end{aligned}$$

In particular, if we assume additionally that  $\nabla \ell$  and  $D^2 g, D^2 f^*$  are Lipschitz, we have the estimate, for  $k \geq 1$ ,

$$\|e_X^k\|^2 + \|e_Y^k\|^2 \leq C\omega^k \quad (37)$$

where  $C$  depends on the Lipschitz constants mentioned above. We observe that we would still obtain a (slower) linear rate if we assumed only Hölder continuity of the same functions.

We deduce that

$$\begin{aligned} \gamma \|X^{k+1} - \hat{X}\|^2 &\leq \frac{1}{\tau} \langle X^k - X^{k+1}, X^{k+1} - \hat{X} \rangle \\ &\quad - \langle Y^k - \hat{Y}, K(X^{k+1} - \hat{X}) \rangle + \langle e_X^{k+1}, X^{k+1} - \hat{X} \rangle, \end{aligned}$$

that is

$$\begin{aligned} \frac{1 + 2\tau\gamma}{2\tau} \|X^{k+1} - \hat{X}\|^2 + \frac{1}{2\tau} \|X^{k+1} - X^k\|^2 &\leq \frac{1}{2\tau} \|X^k - \hat{X}\|^2 \\ &\quad - \langle Y^k - \hat{Y}, K(X^{k+1} - \hat{X}) \rangle + \langle e_X^{k+1}, X^{k+1} - \hat{X} \rangle \end{aligned}$$

and in the same way

$$\begin{aligned} \frac{1 + 2\sigma\delta}{2\sigma} \|Y^{k+1} - \hat{Y}\|^2 + \frac{1}{2\sigma} \|Y^{k+1} - Y^k\|^2 &\leq \frac{1}{2\sigma} \|Y^k - \hat{Y}\|^2 \\ &\quad + \langle Y^{k+1} - \hat{Y}, K(\bar{X}^{k+1} - \hat{X}) \rangle + \langle e_Y^{k+1}, Y^{k+1} - \hat{Y} \rangle. \end{aligned}$$

As in [6] we introduce  $\Delta_k := \frac{1}{2\tau} \|X^k - \hat{X}\|^2 + \frac{1}{2\sigma} \|Y^{k-1} - \hat{Y}\|^2$ . We sum the previous inequalities to find that

$$\begin{aligned} (1 + \mu)\Delta_{k+1} + \frac{1}{2\tau} \|X^{k+1} - X^k\|^2 + \frac{1}{2\sigma} \|Y^k - Y^{k-1}\|^2 \\ \leq \Delta_k - \langle Y^k - \hat{Y}, K(X^{k+1} - X^k) \rangle + \theta \langle (Y^{k-1} - \hat{Y}^k, K(X^k - X^{k-1})) \rangle \\ + \theta \langle Y^k - Y^{k-1}, K(X^k - X^{k-1}) \rangle + \langle e_X^{k+1}, X^{k+1} - \hat{X} \rangle + \langle e_Y^k, Y^k - \hat{Y} \rangle. \end{aligned} \quad (38)$$

The simplest way to get rid of the two error terms (which might not be optimal) is to bound these by  $\|e_X^{k+1}\|^2/(2\gamma) + \|e_Y^k\|^2/(2\delta) + (\mu/2)\Delta_{k+1}$ , which yields

$$\begin{aligned} (1 + \frac{\mu}{2})\Delta_{k+1} + \frac{1}{2\tau} \|X^{k+1} - X^k\|^2 + \frac{1}{2\sigma} \|Y^k - Y^{k-1}\|^2 \\ \leq \Delta_k - \langle Y^k - \hat{Y}, K(X^{k+1} - X^k) \rangle + \theta \langle (Y^{k-1} - \hat{Y}^k, K(X^k - X^{k-1})) \rangle \\ + \theta \langle Y^k - Y^{k-1}, K(X^k - X^{k-1}) \rangle + \frac{1}{2\gamma} \|e_X^{k+1}\|^2 + \frac{1}{2\delta} \|e_Y^k\|^2. \end{aligned} \quad (39)$$

Letting now  $\tilde{\omega} := (1 + \theta)/(2 + \mu/2) \in (\omega, 1)$ , using (37) and adapting the proofs of [6, Thm. 3] [22, Thm. 7], we obtain that  $(X^k, Y^k) \rightarrow (\hat{X}, \hat{Y})$  at rate  $O(\tilde{\omega}^{k/2})$  (which is strictly slower than the rate for  $(x^k, y^k)$ ). As mentioned already, if  $\nabla \ell$ ,  $D^2 g$ ,  $D^2 f^*$  are only Hölder continuous, we may still adapt the above proof to still obtain a (slower) linear convergence rate.

## References

- [1] Rémy Abergel and Lionel Moisan. The Shannon Total Variation. *J. Math. Imaging Vision*, 59(2):341–370, 2017.
- [2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.
- [3] Andrea Braides.  $\Gamma$ -convergence for beginners, volume 22 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2002.
- [4] Corentin Caillaud and Antonin Chambolle. Error estimates for finite differences approximations of the total variation. (preprint hal-02539136), April 2020.
- [5] Antonin Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vision*, 20(1-2):89–97, 2004. Special issue on mathematics and image analysis.
- [6] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40(1):120–145, 2011.
- [7] Antonin Chambolle and Thomas Pock. On the ergodic convergence rates of a first-order primal-dual algorithm. *Mathematical Programming*, pages 1–35, 2015. (online first).
- [8] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numer.*, 25:161–319, 2016.
- [9] Antonin Chambolle and Thomas Pock. Total roto-translational variation. *Numer. Math.*, 142(3):611–666, 2019.
- [10] Antonin Chambolle and Thomas Pock. Crouzeix-Raviart approximation of the total variation on simplicial meshes. *J. Math. Imaging Vision*, 62(6-7):872–899, 2020.
- [11] Antonin Chambolle and Thomas Pock. Finite differences and finite elements discretizations of the total variation. Technical report, CEREMADE, Université Paris-Dauphine-PSL and ICG, TU Graz, 2020. (preprint).
- [12] Laurent Condat. Discrete total variation: new definition and minimization. *SIAM J. Imaging Sci.*, 10(3):1258–1290, 2017.
- [13] Gianni Dal Maso. *An introduction to  $\Gamma$ -convergence*, volume 8 of *Progress in Nonlinear Differential Equations and their Applications*. Birkhäuser Boston, Inc., Boston, MA, 1993.
- [14] Philippe Destuynder, Mohamed Jaoua, and Hela Sellami. A dual algorithm for denoising and preserving edges in image processing. *J. Inverse Ill-Posed Probl.*, 15(2):149–165, 2007.

- [15] Andreas Griewank and Christèle Faure. Piggyback differentiation and optimization. In Lorenz T. Biegler, Matthias Heinkenschloss, Omar Ghattas, and Bart van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, pages 148–164, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [16] Marc Herrmann, Roland Herzog, Stephan Schmidt, José Vidal-Núñez, and Gerd Wachsmuth. Discrete total variation with finite elements and applications to imaging. *Journal of Mathematical Imaging and Vision*, 61(4):411–431, 2019.
- [17] Michael Hintermüller, Carlos N. Rautenberg, and Jooyoung Hahn. Functional-analytic and numerical issues in splitting methods for total variation-based image reconstruction. *Inverse Problems*, 30(5):055014, 34, 2014.
- [18] Karl Kunisch and Thomas Pock. A bilevel optimization approach for parameter learning in variational models. *SIAM J. Imaging Sci.*, 6(2):938–983, 2013.
- [19] Ming-Jun Lai, Bradley Lucier, and Jingyue Wang. *The Convergence of a Central-Difference Discretization of Rudin-Osher-Fatemi Model for Image Denoising*, pages 514–526. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [20] Mahesh Chandra Mukkamala, Peter Ochs, Thomas Pock, and Shoham Sabach. Convex-Concave Backtracking for Inertial Bregman Proximal Gradient Algorithms in Nonconvex Optimization. *SIAM Journal on Mathematics of Data Science*, 2(3):658–682, 2020.
- [21] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms. In *International Conference of Computer Vision (ICCV 2011)*, pages 1762–1769, 2011.
- [22] Julian Rasch and Antonin Chambolle. Inexact first-order primal-dual algorithms. *Comput. Optim. Appl.*, 76(2):381–430, 2020.
- [23] Pierre-Arnaud Raviart and Jean-Marie Thomas. A mixed finite element method for 2nd order elliptic problems. pages 292–315. *Lecture Notes in Math.*, Vol. 606, 1977.
- [24] Jingyue Wang and Bradley J. Lucier. Error bounds for finite-difference methods for Rudin-Osher-Fatemi image smoothing. *SIAM J. Numer. Anal.*, 49(2):845–868, 2011.