Image Deconvolution with the Alternating-Direction Method of Multipliers

Adapted from Gordon Wetzstein's notes by David Lindell for CSC2529 at the University of Toronto

This document serves as a supplement to the material discussed in the lectures. The document is not meant to be a comprehensive review of image deconvolution or iterative optimization, but rather an intuitive introduction to the basic mathematical concepts of non-blind image deconvolution and efficient implementation strategies with the Alternating-Direction Method of Multipliers (ADMM).

To keep this document to a reasonable length, we will make the assumption that our observations are corrupted by zero-mean Gaussian i.i.d. noise and use a single prior for regularizing our objective function.

1 Image Formation

Given a 2D image x and a shift-invariant 2D convolution kernel or point spread function (PSF) c, a 2D image b is formed as

$$b = c * x + \eta. \tag{1}$$

Here, b is the measured image, which is usually blurry, i.e., in most imaging applications the PSF is an optical low-pass filter. The measurements are corrupted by an additive, signal-independent noise term η.

The convolution theorem states that Equation 1 can be similarly written as a multiplication in the Fourier domain:

$$b = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{c\right\} \cdot \mathcal{F}\left\{x\right\}\right\} + \eta, \tag{2}$$

where $\cdot$ is the element-wise product. Note that Equations 1 and 2 are numerically only equivalent when the convolution is performed with circular boundary conditions.

Deconvolution is the problem of finding an estimate $\widetilde{x}$ of the latent image from blurry, possibly also noisy, measurements b.

For the purpose of this set of notes, we will interchangeably use the above "signal processing notation" with x and b describing 2D images and an "algebraic notation" that expresses the same concepts using matrices and vectors:

$$b = c * x \;\Leftrightarrow\; \mathbf{b} = \mathbf{C}\mathbf{x}. \tag{3}$$

We consider these formulations equivalent, where $\mathbf{C} \in \mathbb{R}^{N \times N}$ is a square circulant Toeplitz matrix implementing a shift invariant convolution with convolution kernel c and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^N$ are the vectorized forms of the corresponding images x and b, respectively.

2 Inverse Filtering and Wiener Deconvolution

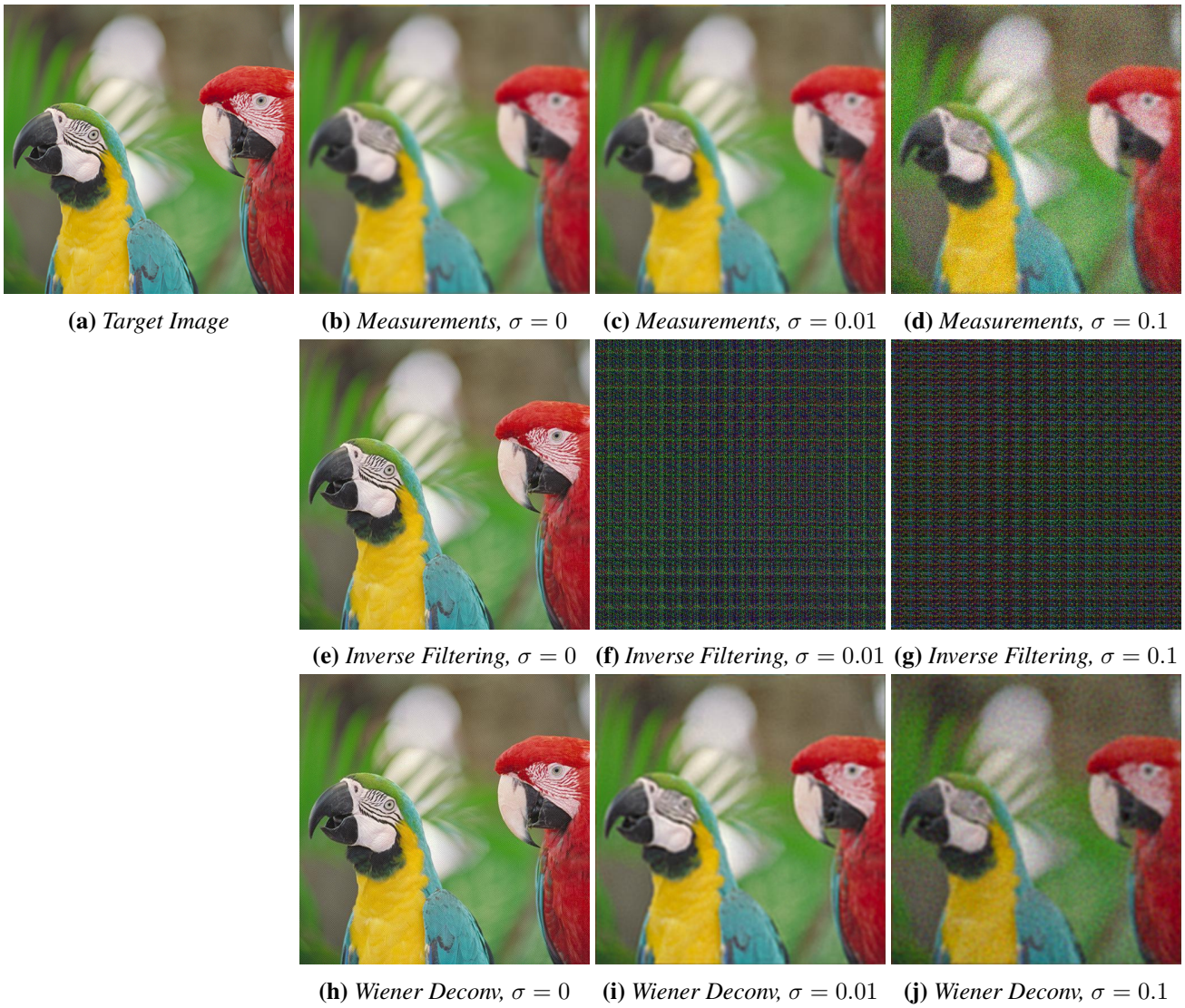
The most straightforward approach of inverting Equation 2 is inverse filtering. For this purpose, a per-image-frequency division of the optical transfer function (OTF) $\mathcal{F}\left\{c\right\}$ is performed as

$$\widetilde{x}_{\text{if}} = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\left\{b\right\}}{\mathcal{F}\left\{c\right\}}\right\}. \tag{4}$$

Although inverse filtering is computationally efficient, it is usually problematic when the values of $\mathcal{F}\left\{c\right\}$ are small, i.e., the convolution kernel has zeros in the Fourier domain. Unfortunately, this is the case for most relevant PSFs in imaging and optics. Divisions by zero or values close to zero will severely amplify measurement noise, as illustrated in Fig. 1.

The primary problem of inverse filtering is that measurement noise is ignored for the reconstruction. Wiener filtering applied to the deconvolution problem adds a damping factor to the inverse filter:

$$\widetilde{x}_{\text{wf}} = \mathcal{F}^{-1}\left\{\frac{\left|\mathcal{F}\left\{c\right\}\right|^2}{\left|\mathcal{F}\left\{c\right\}\right|^2 + \frac{1}{SNR}} \cdot \frac{\mathcal{F}\left\{b\right\}}{\mathcal{F}\left\{c\right\}}\right\}, \tag{5}$$



(a) *Target Image* **(b)** *Measurements,* $\sigma = 0$ **(c)** *Measurements,* $\sigma = 0.01$ **(d)** *Measurements,* $\sigma = 0.1$

(e) *Inverse Filtering,* $\sigma = 0$ **(f)** *Inverse Filtering,* $\sigma = 0.01$ **(g)** *Inverse Filtering,* $\sigma = 0.1$

(h) *Wiener Deconv,* $\sigma = 0$ **(i)** *Wiener Deconv,* $\sigma = 0.01$ **(j)** *Wiener Deconv,* $\sigma = 0.1$

Figure 1: *Inverse filtering and Wiener Deconvolution for the birds image;* σ *is the standard deviation of the zero-mean i.i.d. Gaussian noise added to the corresponding measurements.*

where SNR is the signal-to-noise ratio. If no noise is present in the measurements, the SNR is infinitely high. In that particular case, Wiener filtering is equivalent to inverse filtering. In all other cases, Equation 5 adds a per-frequency damping factor that requires the signal magnitude and the noise power spectral density for each frequency to be known. A common approximation to this is to choose the signal term as the mean image intensity and the noise term as the standard deviation of the (usually zero-mean Gaussian i.i.d.) noise distribution η.

Wiener deconvolution generally achieves reasonable results, as seen in Figure 1. However, perhaps the fundamental problem with these intuitive filter-based approaches to solving the deconvolution problem is that we have no way to include prior knowledge of natural image statistics. State-of-the-art data-driven methods, for example, are extremely powerful, but we simply cannot make use of them with the inverse or Wiener filtering approach. To overcome this challenge, we will use a natural image prior called "total variation" as discussed in the next section. Then we will derive an iterative optimization approach that allows us to combine this and other (data-driven) natural image priors with the image formation model of our deconvolution problem.

3 Natural Image Priors

So, what is a natural image prior? It is simply a mathematical model that tells us which distribution of pixels is more likely than another. When solving ill-posed inverse problems, such as deconvolution, there are infinitely many solutions that could have led to a set of observations. A prior encodes knowledge of what the solution should look like, allowing us to pick a desireable solution among the many feasible solutions that might align with the measurements.

3.1 Motivating Example with Gaussian Noise

Let's see how the idea of a prior distribution falls out of maximum a posteriori (MAP) estimation in the presence of Gaussian noise. Here, we use a generic model for signal-independent, additive noise:

$$\widetilde{\mathbf{b}} = \mathbf{x} + \eta. \tag{6}$$

The noise term η follows a zero-mean i.i.d. Gaussian distribution $\eta_i \sim \mathcal{N}\left(0, \sigma^2\right)$. Conceptually, We can model the noise-free signal $\mathbf{x} \in \mathbb{R}^N$ as a Gaussian distribution with zero variance, i.e. $\mathbf{x}_i \sim \mathcal{N}\left(\mathbf{x}, 0\right)$, which allows us to model $\mathbf{b}$ as a Gaussian distribution. Remember that the sum of two Gaussian distributions $\mathbf{y}_1 \sim \mathcal{N}\left(\mu_1, \sigma_1^2\right)$ and $\mathbf{y}_2 \sim \mathcal{N}\left(\mu_2, \sigma_2^2\right)$ is also a Gaussian distribution $\mathbf{y}_1 + \mathbf{y}_2 \sim \mathcal{N}\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$. Therefore, the noisy intensity measurements $\mathbf{b}_i$ follow a per-pixel normal distribution

$$p\left(\widetilde{\mathbf{b}}_i | \mathbf{x}_i, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(\mathbf{b}_i - \mathbf{x}_i\right)^2}{2\sigma^2}}. \tag{7}$$

Due to the fact that the noise is independent for each pixel, we can model the joint probability of all M pixels as the product of the individual probabilities:

$$p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right) = \prod_{i=1}^{M} p\left(\widetilde{\mathbf{b}}_i | \mathbf{x}_i, \sigma\right) \propto e^{-\frac{\|\mathbf{b} - \mathbf{x}\|_2^2}{2\sigma^2}}. \tag{8}$$

Then we apply Bayes' rule to get an expression for the posterior distribution

$$p\left(\mathbf{x} | \widetilde{\mathbf{b}}, \sigma\right) = \frac{p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right) p\left(\mathbf{x}\right)}{p\left(\mathbf{b}\right)} \propto p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right) p\left(\mathbf{x}\right), \tag{9}$$

where $p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right)$ is the *likelihood* and $p\left(\mathbf{x}\right)$ can be interpreted as a *prior* on the latent image $\mathbf{x}$. The MAP estimate of $\mathbf{x}$ is usually calculated by maximizing the natural logarithm of the posterior distribution or, equivalently, minimizing the negative logarithm of the posterior distribution:

$$\mathbf{x}_{\text{MAP}} = \arg\max_{\{\mathbf{x}\}} \log\left(p\left(\mathbf{x} | \widetilde{\mathbf{b}}, \sigma\right)\right) = \arg\max_{\{\mathbf{x}\}} \log\left(p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right) p\left(\mathbf{x}\right)\right) \tag{10}$$

$$= \arg\min_{\{\mathbf{x}\}} -\log\left(p\left(\widetilde{\mathbf{b}} | \mathbf{x}, \sigma\right)\right) - \log\left(p\left(\mathbf{x}\right)\right) \tag{11}$$

$$= \arg\min_{\{\mathbf{x}\}} \frac{1}{2\sigma^2} \left\| \widetilde{\mathbf{b}} - \mathbf{x} \right\|_2^2 - \log\left(p\left(\mathbf{x}\right)\right) \tag{12}$$

$$= \arg\min_{\{\mathbf{x}\}} \frac{1}{2\sigma^2} \left\| \widetilde{\mathbf{b}} - \mathbf{x} \right\|_2^2 + \Psi\left(\mathbf{x}\right) \tag{13}$$

In the last step we let $-\log\left(p(\mathbf{x})\right) = \Psi(\mathbf{x})$, where Ψ is a regularizer that encodes the prior on $\mathbf{x}$. This derivation may seem simple, but Equation 13 is really important. This is the standard form of the objective function of a Gaussian denoising problem. This derivation provides the direct link between the statistical signal processing and the algebraic, regularized optimization perspectives of solving the denoising problem. Inspecting this derivation also tells us that with our regularized optimization approach, we always estimate the MAP solution to this problem, but we never analyze the posterior distribution. So we pick one solution from an entire distribution, but we have no information about the distribution itself, which could be very interesting!

3.2 Uniform Prior

For a uniform prior on images with normalized intensity values, i.e. $p(\mathbf{x}_i) = \begin{cases} 1 & \text{for } 0 \leq \mathbf{x}_i \leq 1 \\ 0 & \text{otherwise} \end{cases}$, this results in a maximum-likelihood estimation, which is equivalent to the common least-squared error form

$$\mathbf{x}_{flat} = \arg\min_{\{\mathbf{x}\}} \frac{1}{2\sigma^2} \left\| \widetilde{\mathbf{b}} - \mathbf{x} \right\|_2^2 = \widetilde{\mathbf{b}}, \tag{14}$$

resulting in a trivial closed-form solution. This means that denoising an image with additive Gaussian noise requires additional information to be imposed, via the prior $\Psi(\mathbf{x})$.

3.3 Self-similarity Priors

Some of the most popular priors for image denoising are self-similarity priors, such as non-local means (NLM) [Buades et al. 2005] or BM3D [Dabov et al. 2007]. These were already introduced in lecture 4, but formally NLM minimizes the following objective function:

$$\mathbf{x}_{\text{NLM}} = \arg\min_{\{\mathbf{x}\}} \frac{1}{2\sigma^2} \left\| \widetilde{\mathbf{b}} - \mathbf{x} \right\|_2^2 + \Psi(\mathbf{x}) = \text{NLM}\left(\widetilde{\mathbf{b}}, \sigma^2\right). \tag{15}$$

Therefore, NLM (and similarly BM3D) can be interpreted not only as a (nonlinear) filter that remove noise from images, but more generally as maximum a posteriori solutions to the Gaussian image denoising problem outlined above. Fast implementations for both image and video denoising using NLM of grayscale and color images and videos are included in the OpenCV[1] software package. These implementations are recommended and they can be interfaced via OpenCV's Python wrappers (also from Matlab).

Common regularizers include smoothness, sparseness, sparse gradients, self-similarity-promoting regularizers, and many others. For a vectorized image $\mathbf{x}$, a sparseness regularizer on the image itself could be mathematically expressed by the ℓ_1-norm $\|\mathbf{x}\|_1 = \sum |\mathbf{x}_i|$ or a smoothness regularizer as $\|\Delta\mathbf{x}\|_2$, where Δ is the finite difference approximation of the Laplace operator, i.e., the second derivative of the image. In the following, we will focus on one of the most popular regularizers in image processing called total variation (TV) [Rudin et al. 1992].

3.4 Total Variation Prior

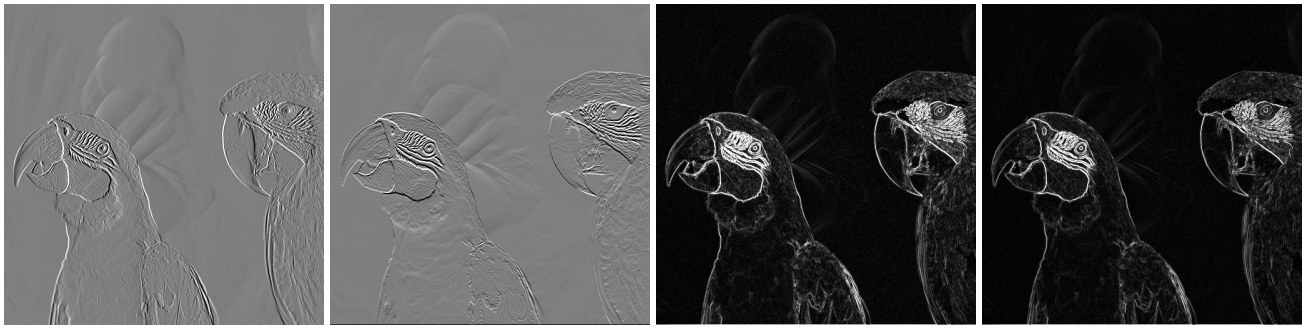
The intuition behind TV is that most natural images can be well modeled by piecewise constant approximations. If you look around you or at photographs of natural scenes, you may notice that many parts of these images are constant over large regions with sharp transitions between different objects. This is exactly what TV attempts to model: sharp edges between areas of constant intensity.

To model TV, we need to compute the first derivative of our image. We denote the first derivative in the horizontal direction as $\mathbf{D}_x\mathbf{x}$ and in the vertical direction as $\mathbf{D}_y\mathbf{x}$. Here, $\mathbf{D}_{x/y} \in \mathbb{R}^{N\times N}$ are matrices encoding the finite differences operator. Again, it is important to keep the duality between the algebraic matrix–vector notation and the signal processing notation in mind, i.e.

$$\mathbf{D}_x\mathbf{x} \;\Leftrightarrow\; x * d_x,\; d_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{D}_y\mathbf{x} \;\Leftrightarrow\; x * d_y,\; d_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{16}$$

Thus, we can equally compute the finite differences derivatives using a matrix–vector multiplication or using a convolution with the kernels d_x and d_y. The resulting image gradients will contain both positive and negative values. We visualize them for our test image in Figs. 2 (a,b), normalized within a range between -0.3 and 0.3 so that gray represents a value of 0.

[1] http://docs.opencv.org/2.4/modules/photo/doc/denoising.html



(a) *Horizontal image gradients* $\mathbf{D}_x\mathbf{x}$ **(b)** *Vertical image gradients* $\mathbf{D}_y\mathbf{x}$ **(c)** *Anisotropic TV* **(d)** *Isotropic TV*

Figure 2: *Image gradients (a,b) as well as isotropic and anisotropic variants of TV without summing over the pixels. Both (c,d) are mostly black, indicating that this image indeed has many zero-valued, i.e., sparse, gradients. For clarify, we show these gradients and TV for the grayscale version of the "birds" image.*

The TV pseudo-norm promotes sparse gradients using an ℓ_1-norm of the image gradients. For this purpose, anisotropic TV simply adds the ℓ_1-norms of both gradients terms as

$$\mathrm{TV}_{\text{anisotropic}}\left(\mathbf{x}\right) = \left\|\mathbf{D}_x\mathbf{x}\right\|_1 + \left\|\mathbf{D}_y\mathbf{x}\right\|_1 = \sum_{i=1}^{N} \left|\left(\mathbf{D}_x\mathbf{x}\right)_i\right| + \left|\left(\mathbf{D}_y\mathbf{x}\right)_i\right| = \sum_{i=1}^{N} \sqrt{\left(\mathbf{D}_x\mathbf{x}\right)_i^2} + \sqrt{\left(\mathbf{D}_y\mathbf{x}\right)_i^2}. \tag{17}$$

The isotropic variant of TV combines the gradients in a slightly different way as

$$\mathrm{TV}_{\text{isotropic}}\left(\mathbf{x}\right) = \left\|\mathbf{D}\mathbf{x}\right\|_{2,1} = \sum_{i=1}^{N} \left\| \begin{bmatrix} \left(\mathbf{D}_x x\right)_i \\ \left(\mathbf{D}_y x\right)_i \end{bmatrix} \right\|_2 = \sum_{i=1}^{N} \sqrt{\left(\mathbf{D}_x\mathbf{x}\right)_i^2 + \left(\mathbf{D}_y\mathbf{x}\right)_i^2}, \tag{18}$$

where $\left\|\cdot\right\|_{2,1}$ is known as the $\ell_{2,1}$-norm.

Here is intuitive explanation of what these two variants of TV do differently: the anisotropic version simply promotes sparse horizontal and vertical gradients, but it doesn't specify that non-zero gradients in both dimensions should ideally be at the same pixel location. So you could end up with a non-zero horizontal gradient at some pixel and the corresponding non-zero vertical gradient at the next pixel over. The isotropic variant does not allow this, because it constraints the magnitude of the two-dimensional image gradient $\sqrt{\left(\mathbf{D}_x\mathbf{x}\right)_i^2 + \left(\mathbf{D}_y\mathbf{x}\right)_i^2}$ to be sparse, rather than its individual components separately.

We visualize both variants of the TV terms without summing over the pixels in Figs. 2 (c,d). Qualitatively, they look very similar. Mathematically, it will be slightly easier to work with the anisotropic variant, but the isotropic variant gives slightly better results in many applications when this is used as a regularization term in an iterative optimization procedure. We will derive both variants in the following and show how to incorporate TV and also other regularizers into iterative solvers for the deconvolution problem.

If you are interested in TV-like regularizers, you may enjoy reading about some generalizations of the conventional TV regularizer discussed here, including hyper-Laplacian priors [Krishnan and Fergus 2009].

4 Regularized Deconvolution with ADMM

In this section, we introduce a powerful and flexible iterative optimization approach for solving the deconvolution problem with priors. To this end, we introduce the Alternating-Direction Method of Multipliers (ADMM) method, which is a general strategy for solving regularized inverse problems. Then we show how to use this general framework to solve the deconvolution problem efficiently using two proximal operators, one for the data fidelity term and another one for the regularizer. The former operator turns out to be an inverse filtering step and we will show

efficient proximal operators for two regularizers: total variation and deep network-based Gaussian denoisers, such as DnCNN [Zhang et al. 2017].

4.1 The Alternating-Direction Method of Multipliers

First, let us introduce ADMM [Boyd et al. 2011] in general terms before applying it to the specific problem of image deconvolution in the next subsection.

Consider a linear image formation model of the form

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}, \tag{19}$$

where $\mathbf{x} \in \mathbb{R}^N$ is a vector of unknowns, $\mathbf{b} \in \mathbb{R}^M$ is a vector containing the observations, $\boldsymbol{\eta}$ is additive noise, and $\mathbf{A} \in \mathbb{R}^{M \times N}$ encodes the linear image formation model.

A general formulation for inverse problems in computational imaging is

$$\underset{\{\mathbf{x}\}}{\text{minimize}} \; \underbrace{\frac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2}_{\text{data fidelity term}} + \underbrace{\lambda \Psi\left(\mathbf{x}\right)}_{\text{regularizer}}. \tag{20}$$

The first term is often referred to as the data fidelity term as it ensures that whatever solution we will find, it should match our observations $\mathbf{b}$ as defined by the image formation model $\mathbf{A}$. $\Psi : \mathbb{R}^n \rightarrow \mathbb{R}$ is the regularization operator modeling prior knowledge of the latent image. The parameter λ defines the relative weight between the data fidelity and the regularization term.

Solving Equation 20 directly by minimizing it via the gradient descent method or other solvers, such as Adam [Kingma and Ba 2014], may not work well. The reason is that these direct approaches may not converge well or it may be challenging to compute the gradients of the objective function in a computationally efficient way. Moreover, changing the regularizer may require us to re-write our solver from scratch. To address these challenges, we will consider an alternative form of Equation 20:

$$\begin{aligned} &\underset{\{x\}}{\text{minimize}} \; \underbrace{\frac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2}_{f(\mathbf{x})} + \underbrace{\lambda \Psi\left(\mathbf{z}\right)}_{g(\mathbf{z})} \\ &\text{subject to } \mathbf{D}\mathbf{x} - \mathbf{z} = 0 \end{aligned} \tag{21}$$

Here, we introduce an additional "slack" variable $\mathbf{z} \in \mathbb{R}^O$. This additional variable allows us to split the two terms of the objective function, i.e., the data fidelity term and the regularization term, such that they do not depend on the same variable. Of course, the constraints link these two variables to make sure they match. For now, let's assume that the matrix $\mathbf{D} \in \mathbb{R}^{N \times O}$ is the identity matrix; so we can ignore it but will get back to it in the next subsection when we discuss the TV regularizer. Clearly, this formulation is equivalent to the original problem in Eq. 20.

A general method for solving such constrained optimization problems is to use the Lagrangian form[2], which is given as

$$L(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f\left(\mathbf{x}\right) + g\left(\mathbf{z}\right) + \mathbf{y}^T\left(\mathbf{D}\mathbf{x} - \mathbf{z}\right), \tag{22}$$

where $\mathbf{y}$ is the dual variable. An optimal value of Equation 21 is achieved when $\nabla_{\mathbf{x},\mathbf{z},\mathbf{y}} L = 0$ (i.e., all partial derivatives are zero). Here, $\nabla_{\mathbf{y}} L = 0$ implies that $\mathbf{D}\mathbf{x} - \mathbf{z} = 0$, ensuring the constraint is satisfied.

[2] see, e.g., `https://en.wikipedia.org/wiki/Lagrange_multiplier`

In practice, we make two adjustments to the Lagrangian to (1) improve convergence properties and (2) to perform some algebraic simplification. First, we add an additional quadratic penalty which lifts some properties required for convergence in the original formulation [Boyd et al. 2011], resulting in the augmented Lagrangian.

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{D}\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{D}\mathbf{x} - \mathbf{z}\|. \tag{23}$$

Here, ρ is called the *penalty parameter*.

Second, with some algebraic manipulation to combine the linear and quadratic terms and scale the dual variable, we arrive at the scaled form of the augmented Lagrangian,

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{D}\mathbf{x} - \mathbf{z} + \mathbf{u}\| - \frac{\rho}{2}\|\mathbf{u}\|, \tag{24}$$

where $\mathbf{u} = \mathbf{y}/\rho$ is the scaled dual variable.

We can now write the expression for ADMM, which is used to solve Equation 21.

while not converged:

$$\mathbf{x} \leftarrow \mathbf{prox}_{f,\rho}(\mathbf{v}) = \underset{\{\mathbf{x}\}}{\arg\min}\, L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \underset{\{\mathbf{x}\}}{\arg\min}\, f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{D}\mathbf{x} - \mathbf{v}\|_2^2, \quad \mathbf{v} = \mathbf{z} - \mathbf{u}, \tag{25}$$

$$\mathbf{z} \leftarrow \mathbf{prox}_{g,\rho}(\mathbf{v}) = \underset{\{\mathbf{z}\}}{\arg\min}\, L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \underset{\{\mathbf{z}\}}{\arg\min}\, g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{v} - \mathbf{z}\|_2^2, \quad \mathbf{v} = \mathbf{D}\mathbf{x} + \mathbf{u}, \tag{26}$$

$$\mathbf{u} \leftarrow \mathbf{u} + \mathbf{D}\mathbf{x} - \mathbf{z} \tag{27}$$

The $\mathbf{x}$ and $\mathbf{z}$-updates are performed with what is known as proximal operators $\mathbf{prox}_{\cdot,\rho}$. While not obvious here, the $\mathbf{u}$-update step follows from applying dual ascent to augmented Lagrangian. The interested reader is referred to Boyd et al. [2011] for more details on proximal operators and dual ascent.

The important insight here is that by performing these $\mathbf{x}$ and $\mathbf{z}$-updates separately, we only ever have to consider either $f(\mathbf{x})$ or $g(\mathbf{z})$ in an update but not both simultaneously. As we will see in the following, this approach leads to a very flexible framework that allows us to easily experiment with different regularizers Ψ when solving an inverse problem. This is also known as a "plug-and-play" approach [Venkatakrishnan et al. 2013], because we can simply mix and match image formation models $\mathbf{A}$ and regularizers Ψ to solve different inverse problems with different regularizers without ever having to re-implement any of the proximal operators twice. Finally, we will see that many proximal operators can be implemented very efficiently, often in closed form; this is another benefit of this approach along with its flexibility.

While ADMM can be applied to solve many different inverse problems, each one using a different matrix $\mathbf{A}$, in the following we will only consider the special case of image deconvolution with circular boundary conditions. In this context, the general matrix $\mathbf{A}$ is going to be the square circulant Toeplitz matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$ implementing a shift invariant convolution of the 2D input image x with convolution kernel c, as discussed in Section 1. So it is important to keep the duality between the signal processing notation and the algebra notation in mind, i.e.,

$$c * x = \mathcal{F}^{-1}\{\mathcal{F}\{c\} \cdot \mathcal{F}\{x\}\} \Leftrightarrow \mathbf{C}\mathbf{x}, \tag{28}$$

$$\mathcal{F}^{-1}\{\mathcal{F}\{c\}^* \cdot \mathcal{F}\{x\}\} \Leftrightarrow \mathbf{C}^T\mathbf{x}, \tag{29}$$

$$\mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{b\}}{\mathcal{F}\{c\}}\right\} \Leftrightarrow \mathbf{C}^{-1}\mathbf{b}, \tag{30}$$

because we will move back and forth between these notations.

Standard Form of ADMM with TV and Denoising Regularizers

Note that the standard form of ADMM, i.e., Equation 21, will depend a bit on which regularizer we will be working with. For the TV regularizer we discussed in the last section, the ADMM objective is formulated as

$$\begin{aligned} &\underset{\{x\}}{\text{minimize}} \ \underbrace{\frac{1}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{b}\right\|_2^2}_{f(\mathbf{x})}+\underbrace{\lambda\left\|\mathbf{z}\right\|_1}_{g(\mathbf{z})} \\ &\text{subject to } \mathbf{D}\mathbf{x}-\mathbf{z}=0, \end{aligned} \tag{31}$$

where $\mathbf{D}=[\mathbf{D}_x^T \ \mathbf{D}_y^T]^T \in \mathbb{R}^{2N\times N}$ encodes the finite differences operator for calculating the gradient in the x and y dimensions of the image. Importantly, $\mathbf{z},\mathbf{u}\in\mathbb{R}^{2N}$ are twice as large as $\mathbf{x}\in\mathbb{R}^N$ in this case, because we need to store a gradient in the x and y directions for each image pixel!

The other, slightly more general case we consider is where we use a general regularizer Ψ that "magically" projects an image onto the feasible set of natural images. We will later see that any Gaussian denoiser can be used for this purpose. The ADMM objective is

$$\begin{aligned} &\underset{\{x\}}{\text{minimize}} \ \underbrace{\frac{1}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{b}\right\|_2^2}_{f(\mathbf{x})}+\underbrace{\lambda\Psi\left(\mathbf{z}\right)}_{g(\mathbf{z})} \\ &\text{subject to } \mathbf{x}-\mathbf{z}=0. \end{aligned} \tag{32}$$

For this case we omit the matrix $\mathbf{D}$ or, similarly, assume that it is the identity matrix $\mathbf{D}=\mathbf{I}$. Importantly, $\mathbf{z},\mathbf{u}\in\mathbb{R}^N$ have the same size as $\mathbf{x}\in\mathbb{R}^N$ in this case!

We will differentiate between these two special cases when deriving an efficient implementation of the x-update, i.e., Equation 25, next.

4.2 Efficient Implementation of the $\mathbf{x}$-Update using Inverse Filtering

For the $\mathbf{x}$-update, we need to derive the proximal operator $\mathbf{prox}_{f,\rho}$, which is the following a quadratic program:

$$\mathbf{prox}_{f,\rho}\left(\mathbf{v}\right)=\underset{\{\mathbf{x}\}}{\arg\min}\ f\left(\mathbf{x}\right)+\frac{\rho}{2}\left\|\mathbf{D}\mathbf{x}-\mathbf{v}\right\|_2^2=\underset{\{\mathbf{x}\}}{\arg\min}\ \frac{1}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{b}\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{D}\mathbf{x}-\mathbf{v}\right\|_2^2, \quad \mathbf{v}=\mathbf{z}-\mathbf{u}, \tag{33}$$

To make it easy to follow the derivation step-by-step, we write the objective function out as

$$\begin{aligned} &\frac{1}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{b}\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{D}\mathbf{x}-\mathbf{v}\right\|_2^2 && (34)\\ =&\frac{1}{2}\left(\mathbf{C}\mathbf{x}-\mathbf{b}\right)^T\left(\mathbf{C}\mathbf{x}-\mathbf{b}\right)+\frac{\rho}{2}\left(\mathbf{D}\mathbf{x}-\mathbf{v}\right)^T\left(\mathbf{D}\mathbf{x}-\mathbf{v}\right) && (35)\\ =&\frac{1}{2}\left(\mathbf{x}^T\mathbf{C}^T\mathbf{C}\mathbf{x}-2\mathbf{x}^T\mathbf{C}^T\mathbf{b}+\mathbf{b}^T\mathbf{b}\right)+\frac{\rho}{2}\left(\mathbf{x}^T\mathbf{D}^T\mathbf{D}\mathbf{x}-2\mathbf{x}^T\mathbf{D}^T\mathbf{v}+\mathbf{v}^T\mathbf{v}\right) \end{aligned}$$

The gradient of this expression is

$$\mathbf{C}^T\mathbf{C}\mathbf{x}-\mathbf{C}^T\mathbf{b}+\rho\mathbf{D}^T\mathbf{D}\mathbf{x}-\rho\mathbf{D}^T\mathbf{v}, \tag{36}$$

which, equated to zero, results in the normal equations that allow us to derive an expression for updating $\mathbf{x}$ as

$$\left(\mathbf{C}^T\mathbf{C}+\rho\mathbf{D}^T\mathbf{D}\right)^{-1}\left(\mathbf{C}^T\mathbf{b}+\rho\mathbf{D}^T\mathbf{v}\right). \tag{37}$$

For the specific case of 2D image deconvolution with circular boundary conditions, the most efficient way of directly solving this equation in closed form is inverse filtering (Eq. 4) via the convolution theorem. For this purpose, we need to find expressions that allows us to express all matrix–vector multiplications as Fourier-domain operations.

4.2.1 Special Case of TV Regularizer

As mentioned before, when using the TV regularizer $\mathbf{D}$ is the finite differences operator. Importantly, both operations $\mathbf{C}\mathbf{x}$ and $\mathbf{D}\mathbf{x} = \begin{bmatrix}\mathbf{D}_x^T & \mathbf{D}_y^T\end{bmatrix}^T \mathbf{x}$ (and also their adjoint operations $\mathbf{C}^T\mathbf{b}$ and $\mathbf{D}^T\mathbf{v} = \begin{bmatrix}\mathbf{D}_x^T & \mathbf{D}_y^T\end{bmatrix} \mathbf{v} = \mathbf{D}_x^T\mathbf{v}_1 + \mathbf{D}_y^T\mathbf{v}_2$) can be expressed as convolutions, i.e., $c * x$ and $d_{x/y} * x$. Therefore, we can write operators for the matrix–vector multiplications in Equation 37 as

$$\left(\mathbf{C}^T\mathbf{C} + \rho\mathbf{D}^T\mathbf{D}\right) \Leftrightarrow \mathcal{F}^{-1}\left\{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{c\right\} + \rho\left(\mathcal{F}\left\{d_x\right\}^* \cdot \mathcal{F}\left\{d_x\right\} + \mathcal{F}\left\{d_y\right\}^* \cdot \mathcal{F}\left\{d_y\right\}\right)\right\}, \tag{38}$$

$$\left(\mathbf{C}^T\mathbf{b} + \rho\mathbf{D}^T\mathbf{v}\right) \Leftrightarrow \mathcal{F}^{-1}\left\{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{b\right\} + \rho\left(\mathcal{F}\left\{d_x\right\}^* \cdot \mathcal{F}\left\{v_1\right\} + \mathcal{F}\left\{d_y\right\}^* \cdot \mathcal{F}\left\{v_2\right\}\right)\right\}, \tag{39}$$

where $\mathcal{F}\left\{c\right\}^*$ is the element-wise complex conjugate of $\mathcal{F}\left\{c\right\}$ and $\cdot$ is the element-wise product. This gives rise to the inverse filtering proximal operator, which applies only Fourier transforms and element-wise multiplications or divisions to solve Equation 33 in closed form as

$$\mathbf{prox}_{\|\cdot\|_2,\rho}\left(\mathbf{v}\right) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{b\right\} + \rho\left(\mathcal{F}\left\{d_x\right\}^* \cdot \mathcal{F}\left\{v_1\right\} + \mathcal{F}\left\{d_y\right\}^* \cdot \mathcal{F}\left\{v_2\right\}\right)}{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{c\right\} + \rho\left(\mathcal{F}\left\{d_x\right\}^* \cdot \mathcal{F}\left\{d_x\right\} + \mathcal{F}\left\{d_y\right\}^* \cdot \mathcal{F}\left\{d_y\right\}\right)}\right\}. \tag{40}$$

Just like inverse filtering, the $\mathbf{x}$-update itself may be unstable w.r.t. noise and zeros in the OTF, but embedded in the ADMM iterations this will not affect the resulting estimate of $\mathbf{x}$. Note that all parts of Equation 40 that do not depend on z can be precomputed, i.e., the first part of the nominator and the entire denominator, and do not have to be re-computed throughout the ADMM iterations.

4.2.2 Special Case of Denoising Regularizer

Also as mentioned before, when using a more general regularizer, we assume that $\mathbf{D} = \mathbf{I}$ can be ignored. Therefore, we can write the inverse filtering proximal operator as

$$\mathbf{prox}_{\|\cdot\|_2,\rho}\left(\mathbf{v}\right) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{b\right\} + \rho\mathcal{F}\left\{v\right\}}{\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{c\right\} + \rho}\right\}. \tag{41}$$

Again, all parts of Equation 41 that do not depend on z can be precomputed, which includes the terms $\mathcal{F}\left\{c\right\}^* \cdot \mathcal{F}\left\{b\right\}$ and the entire denominator.

4.3 Updating z with the TV Regularizer

Anisotropic TV Norm In the $\mathbf{z}$-update (Eq. 26), the ℓ_1-norm is convex but not differentiable. Nevertheless, a closed-form solution for the proximal operator exists, such that

$$\mathbf{prox}_{\|\cdot\|_1,\rho}\left(\mathbf{v}\right) = \mathcal{S}_{\lambda/\rho}\left(\mathbf{v}\right) = \underset{\{\mathbf{z}\}}{\arg\min}\ \lambda\left\|\mathbf{z}\right\|_1 + \frac{\rho}{2}\left\|\mathbf{v} - \mathbf{z}\right\|_2^2, \tag{42}$$

with $\mathbf{v} = \mathbf{D}\mathbf{x} + \mathbf{u}$ and $\mathcal{S}_\kappa\left(\cdot\right)$ being the element-wise soft thresholding operator

$$\begin{aligned}\mathcal{S}_\kappa\left(v\right) &= \begin{cases} v - \kappa & v > \kappa \\ 0 & |v| \leq \kappa \\ v + \kappa & v < -\kappa \end{cases} \\ &= \left(v - \kappa\right)_+ - \left(-v - \kappa\right)_+\end{aligned} \tag{43}$$

that can be implemented very efficiently. This is the proximal operator for the anisotropic TV norm.

Isotropic TV Norm Instead of the ℓ_1-norm, the isotropic case uses the sum of the ℓ_2-norms of the finite differences approximation of the horizontal and vertical image gradients as a regularizer. If we choose $\mathbf{z} \in \mathbb{R}^{2\times N}$ so that $\mathbf{z} = \left[\mathbf{D}_x x \quad \mathbf{D}_y x\right]^T$, we can use the $\ell_{2,1}$-norm to write the isotropic version of the regularizer as

$$g\left(\mathbf{z}\right) = \lambda\left\|\mathbf{z}\right\|_{2,1} = \lambda\sum_{i=1}^{N}\left\|\begin{bmatrix}\left(\mathbf{D}_x x\right)_i \\ \left(\mathbf{D}_y x\right)_i\end{bmatrix}\right\|_2 = \lambda\sum_{i=1}^{N}\sqrt{\left(\mathbf{D}_x\mathbf{x}\right)_i^2 + \left(\mathbf{D}_y\mathbf{x}\right)_i^2}, \tag{44}$$

which is also known as the group lasso.

Using the same notation as in previous subsection, the deconvolution problem with an isotropic TV regularizer is formulated as

$$\begin{aligned} \underset{\{x\}}{\text{minimize}} \; & \underbrace{\frac{1}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{b}\right\|_2^2}_{f(\mathbf{x})} + \underbrace{\lambda \sum_{i=1}^{N} \left\| \begin{bmatrix} z_i \\ z_{i+N} \end{bmatrix} \right\|_2}_{g(\mathbf{z})} \\ & \text{subject to } \mathbf{D}\mathbf{x}-\mathbf{z}=0 \end{aligned} \tag{45}$$

where z_i is the i-th element of $\mathbf{z}$. For $1 \leq i \leq N$ it is meant to represent the finite differences approximation in horizontal direction, $(\mathbf{D}_x x)_i$, and for $N+1 \leq i \leq 2N$ the finite differences approximation in vertical direction, $(\mathbf{D}_x y)_i$. Notice that if we replace the ℓ_2-norm in $g(\mathbf{z})$ with the ℓ_1-norm, then we get $\sum_{i=1}^{N} \left\|(z_i, z_{i+N})\right\|_1$ which reduces to $\left\|\mathbf{z}\right\|_1$ and we recover the anisotropic case.

Since only the regularization term $g(\mathbf{z})$ changed as compared to the anisotropic case, the $\mathbf{x}$-update rule for ADMM stays the same as above and only the $\mathbf{z}$-update changes to

$$\mathbf{z} \leftarrow \mathbf{prox}_{\|\cdot\|_{2,1},\rho}\left(\mathbf{v}\right) = \underset{\{\mathbf{z}\}}{\arg\min}\, \lambda \sum_{i=1}^{N} \left\| \begin{bmatrix} z_i \\ z_{i+N} \end{bmatrix} \right\|_2 + \frac{\rho}{2}\left\|\mathbf{v}-\mathbf{z}\right\|_2^2, \quad \mathbf{v}=\mathbf{D}\mathbf{x}+\mathbf{u} \tag{46}$$

The corresponding proximal operator of $g(\mathbf{z})$, the group lasso, is block soft thresholding.

The $\mathbf{z}$-update rule then becomes

$$\begin{bmatrix} z_i \\ z_{i+N} \end{bmatrix} \leftarrow \mathcal{S}_{\lambda/\rho}\left(\begin{bmatrix} v_i \\ v_{i+N} \end{bmatrix} \right), \quad 1 \leq i \leq N \tag{47}$$

where $\mathcal{S}_\kappa\left(\cdot\right)$ is the vector soft-thresholding operator

$$\mathcal{S}_\kappa\left(\mathbf{a}\right) = \left(1-\frac{\kappa}{\left\|\mathbf{a}\right\|_2}\right)_+ \mathbf{a} \tag{48}$$

4.4 Updating z with DnCNN or any Gaussian Denoiser as the Regularizer

Let's closely examine the z-update in Equation 26 once more without considering the matrix $\mathbf{D}$:

$$\begin{aligned} & \underset{\{\mathbf{z}\}}{\arg\min}\, g\left(\mathbf{z}\right) + \frac{\rho}{2}\left\|\mathbf{v}-\mathbf{z}\right\|_2^2, \quad \mathbf{v}=\mathbf{x}+\mathbf{u}, \\ = \; & \underset{\{\mathbf{z}\}}{\arg\min}\, \lambda\Psi\left(\mathbf{z}\right) + \frac{\rho}{2}\left\|\mathbf{v}-\mathbf{z}\right\|_2^2, \\ = \; & \underset{\{\mathbf{z}\}}{\arg\min}\, \Psi\left(\mathbf{z}\right) + \frac{\rho}{2\lambda}\left\|\mathbf{v}-\mathbf{z}\right\|_2^2. \end{aligned} \tag{49}$$

Surprisingly, Equation 49 describes a denoising problem! Recall from Equation 13 that denoising problems for zero-mean Gaussian noise with variance σ^2 are formulated as

$$\underset{\{\mathbf{z}\}}{\arg\min}\, \Psi\left(\mathbf{z}\right) + \frac{1}{2\sigma^2}\left\|\mathbf{v}-\mathbf{z}\right\|_2^2, \tag{50}$$

where $\mathbf{v} \in \mathbb{R}^N$ are the noisy observation of the unknown but noise-free image $\mathbf{z}$. Eqs. 49 and 50 are equivalent assuming that $\sigma^2 = \lambda/\rho$.

The important insight is that we can use any Gaussian denoiser $\mathcal{D} : \mathbb{R}^N \rightarrow \mathbb{R}^N$, including "hand-crafted" algorithms like non-local means (NLM) or BM3D as well as learned CNN-based denoisers, such as DnCNN, as our proximal operator implementing the z-update. This is important, because it allows us to solve virtually any inverse problem in computational imaging using the ADMM updates with any (CNN-based) denoiser $\mathcal{D}$ applied with (or pre-trained for) a noise variance of σ^2—that is, we never have to learn a regularizer that is specific to a particular problem but can always use the same pre-trained denoiser as

$$\mathbf{prox}_{\mathcal{D},\rho}\left(\mathbf{v}\right) = \mathcal{D}\left(\mathbf{v}, \sigma^2 = \frac{\lambda}{\rho}\right). \tag{51}$$

4.5 Pseudo Code

Algorithms 1 and 2 outline pseudo code for deconvolution with ADMM using TV and general denoising regularizers, respectively.

Algorithm 1 ADMM for deconvolution with TV regularizer

initialize ρ and λ
$x = zeros\left(W, H\right);$
$z = zeros\left(W, H\right);$
$u = zeros\left(W, H\right);$
for $k = 1$ **to** max_iters **do**
  $v = z - u$
  $x = \mathbf{prox}_{\|\cdot\|_2,\rho}\left(v\right) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{c\}^* \cdot \mathcal{F}\{b\} + \rho(\mathcal{F}\{d_x\}^* \cdot \mathcal{F}\{v_1\} + \mathcal{F}\{d_y\}^* \cdot \mathcal{F}\{v_2\})}{\mathcal{F}\{c\}^* \cdot \mathcal{F}\{c\} + \rho(\mathcal{F}\{d_x\}^* \cdot \mathcal{F}\{d_x\} + \mathcal{F}\{d_y\}^* \cdot \mathcal{F}\{d_y\})}\right\}$
  $v = \mathbf{Dx} + \mathbf{u}$
  $z = \mathbf{prox}_{\|\cdot\|_1,\rho}\left(v\right) = \mathcal{S}_{\lambda/\rho}\left(v\right)$
  $u = u + \mathbf{Dx} - z$
end for

Algorithm 2 ADMM for deconvolution with denoising regularizer

initialize ρ and λ
$x = zeros\left(W, H\right);$
$z = zeros\left(W, H\right);$
$u = zeros\left(W, H\right);$
for $k = 1$ **to** max_iters **do**
  $v = z - u$
  $x = \mathbf{prox}_{\|\cdot\|_2,\rho}\left(v\right) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{c\}^* \cdot \mathcal{F}\{b\} + \rho\mathcal{F}\{v\}}{\mathcal{F}\{c\}^* \cdot \mathcal{F}\{c\} + \rho}\right\}$
  $v = x + u$
  $z = \mathbf{prox}_{\mathcal{D},\rho}\left(v\right) = \mathcal{D}\left(v, \sigma^2 = \frac{\lambda}{\rho}\right)$
  $u = u + x - z$
end for

4.6 Experiments and Results

Figure 3 shows an example scene along with simulated blurry and noisy measurements. When comparing different approaches to solving the inverse deconvolution problem, we see that ADMM combined with a DnCNN regularizer achieves the best results in this case as measured quantitatively using the peak signal-to-noise ratio (PSNR). The difference between Adam and ADMM with the same TV regularizer are not as small as one may think. This is surprising, because both of these approaches solve the same objective function but ADMM just seems to converge better. Spending more effort on tuning hyperparameters, such as the learning rate of Adam, may make this gap smaller, however. The quantitative difference between ADMM+TV and DMM+DnCNN are perhaps not as big as

(a) *Target Image* **(b)** *Measurements,* $\sigma = 0.1$ **(c)** *Wiener Deconv, PSNR: 19.5 dB, SSIM 0.83*

(d) *Adam+TV, PSNR 26.1 dB* **(e)** *ADMM+TV, PSNR: 26.3 dB* **(f)** *ADMM+DnCNN, PSNR: 26.7 dB*

Figure 3: *Results. A target image (a) is degraded by blurring it and corrupting the measurements with additive Gaussian noise with* $\sigma = 0.1$ *(b). Conventional approaches, including Wiener deconvolution fail at estimating a high-quality reconstruction (c). The Adam solver of the PyTorch package combined with an isotropic TV regularizer does a much better job at recovering the image (d). The ADMM solver with a TV regularizer achieves somewhat comparable results to Adam (e). However, ADMM is flexible in allowing arbitrary denoisers to be used as regularizers, such as DnCNN. For this example, ADMM with DnCNN (f) achieves the best results both qualitative and quantitatively, as measured by the peak signal-to-noise ratio (PSNR).*

one may expect. Qualitatively, the TV norm creates sharp edges but also usually results in a "patchy" appearance, which is promoted through the piecewise constant intensity regularizer and expected. The DnCNN regularizer can reduce noise and patches in the estimated image and this result looks like the most natural of all of these reconstructions. Yet, there is no magic in any of these approaches as none can recover the high-frequency details of the target image that are irreversibly lost by the blur kernel, which acts as a low-pass filter. The only way to "restore" them is to hallucinate them, for example using a generative adversarial network (GAN).

5 Outlook on Unrolled Optimization with Learned Priors

Typically, the ADMM method discussed in the previous sections is implemented by first fixing the hyperparameters λ and ρ and then running the x, z-updates (Eqs. 25,26) either for a fixed number of iterations or until some convergence criteria is satisfied (e.g., the relative change of the residual between two successive iterations is below some threshold). This works really well in most cases. Yet, a really clever insight is that these repetitive iterations, when used with a differentiable prior such as DnCNN, can be ran for a fixed number of iterations and interpreted as a single neural network with the blurry and noisy measurements as input and the estimated latent image as output. This approach is known as unrolled optimization with deep priors [Diamond et al. 2017]. Some of the benefits of

such an unrolled approach include:

- The entire iterative optimization pipeline becomes end-to-end differentiable.
- The hyperparameters λ and ρ can be learned end to end and optimized per iteration. This would, for example, allow ρ to gradually increase over time. Writing a training curriculum for these parameters manually is really tricky, so learning them can be very helpful.
- The denoising prior could be learned or refined end to end as part of the unrolled network. This allows the denoiser to adapt to the specific matrix $\mathbf{A}$ or convolution kernel c. Moreover, if sufficient memory is available one can also learn independent denoising priors in each iteration, allowing the prior to change throughout the iteration procedure.
- Finally, one can include skip connections from any iteration to any other iteration, turning a sequential iterative optimization strategy into recurrent or DenseNet-like structure where earlier iterations can inform later ones through skip connections.

The unrolled optimization strategy provides a formal link between traditional iterative optimization and modern deep learning frameworks, allowing for the best of both worlds to be seamlessly merged. You can find more information on this topic in the paper by Diamond et al. [2017]. This general topic would lend itself well for a possible course project.

References

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3*, 1, 1–122.

BUADES, A., COLL, B., AND MOREL, J. M. 2005. A non-local algorithm for image denoising. In *Proc. IEEE CVPR*, vol. 2, 60–65.

DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing 16*, 8, 2080–2095.

DIAMOND, S., SITZMANN, V., HEIDE, F., AND WETZSTEIN, G. 2017. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*.

KINGMA, D. P., AND BA, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

KRISHNAN, D., AND FERGUS, R. 2009. Fast image deconvolution using hyper-laplacian priors. *Advances in neural information processing systems 22*, 1033–1041.

RUDIN, L., OSHER, S., AND FATEMI, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1–4, 259 – 268.

VENKATAKRISHNAN, S. V., BOUMAN, C. A., AND WOHLBERG, B. 2013. Plug-and-play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing*, 945–948.

ZHANG, K., ZUO, W., CHEN, Y., MENG, D., AND ZHANG, L. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process. 26*, 7, 3142–3155.