Assignment 1/2

Meier Raphael
08-172-926

1 Introduction

The existence of image noise is inherently connected to the acquisition and transmission of the image itself. Hence, image noise is ubiquitous and occurs in diverse image data such as digital photographs, medical images and videos. However, the basic characteristics of image noise can be described in a general way via a noise model (e.g. Gaussian noise or Rayleigh noise) and do not require the adaptation to a specific domain (e.g medical imaging).

Image denoising can be seen as a special case of image restoration. In image restoration, we consider the degraded (noisy) image to be a result of a degradation process [1]. This process is modelled by a degradation function H and an additive noise term η. In this work, we consider (discrete) two-dimensional images and attempt to solve the image denoising problem in the spatial domain. Furthermore, we regard H to be a linear, shift-invariant operator. Hence, the degraded image $g(x, y)$ is given by:

$$g(x, y) = h(x, y) * u(x, y) + \eta(x, y) \tag{1}$$

where $u(x, y)$ is the original, undegraded image, $h(x, y)$ the spatial representation of the degradation function and $*$ indicates the convolution operation. The goal of image restoration is to recover $u(x, y)$, given $g(x, y)$ and some knowledge about the degradation process (H and η). This is realized by an estimation of a function $\tilde{u}(x, y)$ which is as close as possible to $u(x, y)$. Now, image denoising is a special case, where $h(x, y) = 1$ such that equation (1) reduces to:

$$g(x, y) = u(x, y) + \eta(x, y). \tag{2}$$

As already indicated image noise is injected during image acquisition and/or transmission. During acquisition a main source for image noise is the sensor

itself. Sensor noise is caused by poor illumination and/or high temperature, and electronic circuit noise [1]. This type of image noise can be well approximated with a Gaussian noise model, where the noise is assumed to be normally distributed, i.e. $\eta \sim \mathcal{N}(\mu, \sigma)$ with mean μ and standard deviation σ. The aim of this work is to restore an image subject to Gaussian noise.

2 Methods

We denote the two-dimensional image domain by Ω. The intensity value of a pixel at position $(x, y) \in \Omega$ is given by $u(x, y)$, i.e. $u : \Omega \to \mathbb{R}$.

2.1 Total Variation

A common assumption in image denoising is that spurious (noisy) parts of an image show a high total variation. For a continous, two-dimensional image u the total variation is given by

$$TV(u) = \iint_{\Omega} \|\nabla u(x, y)\| \, \mathrm{d}x\mathrm{d}y \tag{3}$$

where $\|\cdot\|$ denotes the ℓ_2-norm. Based on the fact that natural images are well-approximated by piece-wise constant functions, total variation denoising aims at the minimization of the total variation in a given image while guaranteeing that the estimate is sufficiently *close* to the observed noisy image. Thus, a noise-free image $\tilde{u}$ can be estimated by minimizing the following energy functional:

$$\tilde{u} = \arg\min_{u} E(u) \tag{4}$$

where

$$E(u) = \frac{\lambda}{2} \|u - g\|^2 + TV(u). \tag{5}$$

Closeness to the observed image g is measured in a least-squares sense. The trade-off between the least-squares penalty and total variation regularization is controlled by $\lambda > 0$.

2.2 Discretization and Optimization

We consider a spatial discretization of Ω with a constant step size $\Delta = 1$. The total number of pixels in u is given by $M \times N$. Consequently, the energy $E(u)$

can be approximated by the use of finite differences. Using a forward difference scheme, equation (5) can be approximated by:

$$E(u) = \frac{\lambda}{2} \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} (u[i,j] - g[i,j])^2 + \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} \tau[i,j] \tag{6}$$

where

$$\tau[i,j] = \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2}. \tag{7}$$

The corresponding gradient of the energy with respect to u at pixel (i,j) is given by:

$$\begin{aligned} (\nabla_u E(u))_{i,j} &= \lambda(u[i,j] - g[i,j]) + \frac{2u[i,j] - u[i+1,j] - u[i,j+1]}{\tau[i,j]} \\ &\quad + \frac{u[i,j] - u[i-1,j]}{\tau[i-1,j]} + \frac{u[i,j] - u[i,j-1]}{\tau[i,j-1]} \end{aligned} \tag{8}$$

For the derivation of the gradient, we refer the reader to the appendix A. Notice that, the gradient is not defined for the case where $\tau = 0$. By approximating the total variation via $\tau[i,j] = \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2 + \epsilon}$ using a small $\epsilon > 0$, we try to resolve this issue.

Given the convexity of $E(u)$ (on u) and the definition of the gradient, we can minimize $E(u)$ with respect to u using a standard gradient descent approach. We iteratively minimize $E(u)$ by advancing the solution into the direction of steepest descent defined by the negative gradient. We iterate for a predefined number of iterations T and check convergence via the condition $\|\nabla_u E(u)\| > \epsilon$. The step size s is determined via a backtracking line search based on the Armijo rule [4]. The algorithm is listed below.

Algorithm 1 Gradient descent optimization of $E(u)$

```
t := 0, α := 0.05, β := 0.05
while (t < T) ∧ (‖∇_u E(u_t)‖ > ε) do
    compute ∇_u E(u_t)
    s := 1
    while (E(u_t − s · ∇_u E(u_t)) > E(u_t) − α · s · (∇_u E(u_t)^T ∇_u E(u_t))) do
        s := β · s
    end while
    u_{t+1} := u_t − s · ∇_u E(u_t)
    t := t + 1
end while
```

2.3 Implementation

When implementing the gradient descent algorithm derived before, we need to consider two crucial aspects. The definition of appropriate boundary conditions and the computation of finite differences. We chose a Neumann boundary condition for pixels outside the image domain, i.e. $u'(x, y) = 0, \forall (x, y) \notin \Omega$. This corresponds to a constant border padding of the respective image. Finite difference computations were realized via matrix multiplications. The forward difference in x-direction is then given by:

$$u_x = D_{f,x} u \tag{9}$$

with

$$D_{f,x} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & & & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & & & -1 & 1 & 0 \\ 0 & & & & -1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{M \times M}. \tag{10}$$

Notice that, u is a $M \times N$-matrix. The Neumann boundary condition is incorporated by setting all entries of the last row of $D_{f,x}$ to zero. Analogously, we have the forward difference in y-direction:

$$u_y = u D_{f,y} \tag{11}$$

with

$$D_{f,y} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -1 & & & & 0 \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & & \ddots & -1 & & 0 \\ 0 & & & 1 & -1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{N \times N}. \tag{12}$$

The Neumann boundary condition is reflected in the last column of $D_{f,y}$ where all elements are equal to zero. Furthermore, we can see that $D_{f,x} = D_{f,y}^T$. When we inspect the gradient (equation (8)), we see that the numerators of the third and fourth term are in fact backward differences. As a consequence, we define the matrices for backward differences to be:

$$D_{b,x} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & & & & 0 \\ 0 & -1 & 1 & & & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & & & -1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$
$$D_{b,y} = \begin{bmatrix} 0 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & & & 0 \\ 0 & & 1 & \ddots & & \vdots \\ \vdots & & & \ddots & -1 & 0 \\ 0 & & & & 1 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{13}$$

Armed with these definitions, we can rewrite the gradient of the energy (equation (8)) in the following manner:

$$(\nabla_u E(u))_{i,j} = \lambda(u[i,j] - g[i,j]) - \frac{(D_{f,x}u)_{i,j} + (uD_{f,y})_{i,j}}{\tau[i,j]} + \frac{(D_{b,x}u)_{i,j}}{\tau[i-1,j]} + \frac{(uD_{b,y})_{i,j}}{\tau[i,j-1]}. \tag{14}$$

The algorithm was implemented using MATLAB, where above matrices were defined using `spdiags` due to their sparse nature.

2.4 Primal-Dual Formulation

The energy functional for total variation image denoising, given by equation (5), can be seen as a particular case of the following, more general convex optimization problem:

$$\min_{x \in X} F(Kx) + G(x) \tag{15}$$

where F and G are convex functions and K is a linear operator. For the present case, we can see that $F(Kx) = TV(u)$ with $K = \nabla$ and $G(x) = \frac{\lambda}{2} \|u - g\|^2$. This type of (primal) problems are particularly challenging to solve due to the coupling induced by K. However, convex analysis offers a tool for removing the variable coupling – the Legendre-Fenchel transform. More specifically, we take the Legendre-Fenchel transform F^* of $F(Kx)$ and use the fact that $F(Kx) = \sup_y \langle Kx, y \rangle - F^*$ to arrive at a primal-dual reformulation:

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle - F^*(y) + G(x). \tag{16}$$

Above problem is a saddle-point problem which means that min and max are commutative. The primal-dual formulation of our image denoising problem can be obtained in the same manner by taking the Legendre-Fenchel transform of the total variation regularization. The convex conjugate of the ℓ_2-norm corresponds to the unit ball indicator function:

$$\delta(p) = \begin{cases} 0, & \text{if } \|p\| \leq 1 \\ \infty, & \text{otherwise} \end{cases} \tag{17}$$

Consequently, we obtain the primal-dual form:

$$\min_u \max_p \langle p, \nabla u \rangle - \delta(p) + \frac{\lambda}{2} \|u - g\|^2 . \tag{18}$$

The primal-dual problem (equation (16)) can be solved by updating the primal and dual variables in an iterative fashion. The update steps are defined as:

$$y^{t+1} = \text{prox}_{\sigma F^*}(y^t + \sigma K \bar{x}^t) \tag{19}$$

$$x^{t+1} = \text{prox}_{\tau G}(x^t - \tau K^* y^{t+1}) \tag{20}$$

$$\bar{x}^{t+1} = x^{t+1} + \theta(x^{t+1} - x^t) \tag{21}$$

This is an accelerated version of the basic proximal gradient method, where the third step leads to an improved convergence rate [5]. For guaranteeing convergence, the involved parameters have to be chosen appropriately, i.e. $\theta \in (0, 1]$ and $\tau\sigma \left\|K\right\|^2 < 1$ has to hold. Applied to our problem of total variation image denoising, this leads to the following algorithm:

Algorithm 2 Primal-dual optimization of $E(u)$

$t := 0, \left\|K\right\|^2 := 8, \tau := 0.02, \sigma := \frac{0.95}{\left\|K\right\|^2 \tau}, \bar{u}^0 := u^0, p^0 := \nabla u^0$
while $(t < T) \wedge \frac{\left\|u^{t+1} - u^t\right\|}{\left\|u^t\right\|} < \epsilon$ **do**
  $p^{t+1} := \frac{p^n + \sigma \nabla \bar{u}^t}{\max\{1, \left\|p^n + \sigma \nabla \bar{u}^t\right\|\}}$
  $u^{t+1} := \frac{u^t + \tau \nabla \cdot p^{t+1} + \tau \lambda g}{1 + \tau \lambda}$
  $\bar{u}^{t+1} := u^{t+1} + \theta(u^{t+1} - u^t)$
  $t := t + 1$
end while

In our implementation of the primal dual algorithm, the gradient and divergence operators are discretized in the exact same manner as before, i.e. using finite difference approximations for derivatives and Neumann boundary conditions. Our choice for $\left\|K\right\|^2$ is based on the fact that it can be shown that $\left\|K\right\|^2 = \left\|\nabla\right\|^2 \leq \frac{8}{\Delta^2}$ ([2], since we have $\Delta = 1$ we set $\left\|K\right\|^2 = 8$). Moreover, we decided to employ a rather simple convergence criterium $\frac{\left\|u^{t+1} - u^t\right\|}{\left\|u^t\right\|} < \epsilon$. Notice that, $p^t \in \mathbb{R}^{M \times N \times 2}$ and $u^t \in \mathbb{R}^{M \times N}$. The update step for the primal variables computes actually the divergence of the gradient which corresponds to the laplacian of the current dual estimate.

3 Results

3.1 Algorithm 1 – Gradient descent

We evaluated our method on a two-dimensional gray-value image. Gaussian noise with $\mu = 0$ and $\sigma = 0.1$ has been added to the image prior to denoising. The original (i.e. ground truth) and noisy images are shown in figure 1. In figure 2, the progression of the denoising is shown for different iterations, starting with the initial estimate and ending with the image obtained at convergence. Furthermore, we show images obtained for very low ($\lambda = 0.1$), intermediate (optimal) ($\lambda = 15$) and very high ($\lambda = 50$) values of lambda (figure 3).

Finally, we performed a gridsearch for different values ($\{0, 0.1, 1, 5, 10, 15, 20, 25, 30, 35, 40\}$)

of λ to estimate the optimal value with respect to the error between the restored image $\tilde{u}$ and the ground truth image u. The error was quantified via the sum of squared differences: $SSD = \sum_i \sum_j (\tilde{u}(i,j) - u(i,j))^2$. When we plot values for

Fig. 1: Left: Original gray-value image. Right: Image with added Gaussian noise ($\sigma = 0.1$).

Fig. 2: Estimated $\tilde{u}$ (gradient descent) for iterations $t = 1$, $t = 5$, $t = 10$, $t = 100$ and when convergence is achieved (from top left to bottom right image).

Fig. 3: Left: $\lambda = 0.1$. Middle: $\lambda = 15$. Right: $\lambda = 50$.

SSD against λ in figure 4, we can see that the optimal value for λ corresponds to 15. All experiments have been performed with a maximum number of iterations $T = 1000$ and backtracking line search parameters $\alpha = 0.05$ and $\beta = 0.05$ (empirically chosen).

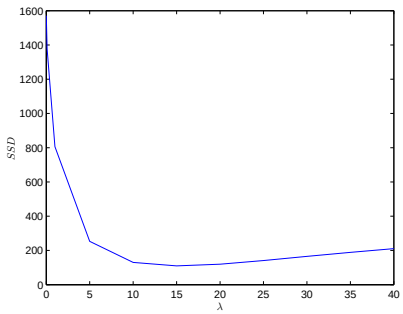

Fig. 4: Plot of SSD vs. λ. Minima of $SSD = 111.33$ for $\lambda = 15$.

3.2 Algorithm 2 – Primal-Dual method

We employed the same experimental setting to evaluate our primal-dual algorithm for solving the total variation image denoising problem. We chose $\tau = 0.02$, $\sigma = \frac{0.95}{\|K\|^2 \tau}$ for all experiments (empirically chosen). In figure 5, the progression of the denoising is shown for different iterations, starting with the initial estimate and ending with the image obtained at convergence. Note that the convergence of the primal-dual algorithm seems to be faster and for the present choice of parameters and convergence criterion is usually satisfied at around 70 iterations (compared to over 500 iterations for the gradient descent). This is also appearant when comparing the level of noise present in the image for each of the method for $t = 5$.

Moreover, we also performed a gridsearch for the same range of values of λ as before. When we plot values for SSD against λ in figure 6, we can see that the optimal value for λ corresponds again to 15. All experiments have been performed with a maximum number of iterations $T = 1000$.

4 Discussion

From our results, we can clearly recognize that the proposed denoising method preserves edge information in the image. However, the method suffers from the introduction of staircase-artifacts. Extensions, such as e.g. Adaptive Total Variation regularization [3], have been proposed to resolve this problem. Furthermore, we can see that λ is responsible for the trade-off between the least-squares penalty and total variation regularization. A very low λ corresponds to a strong regularization. Thus, we downweight the correspondence between our

Fig. 5: Estimated $\tilde{u}$ (primal-dual method) for iterations $t = 1$, $t = 5$, $t = 10$, $t = 50$ and when convergence is achieved (from top left to bottom right image).

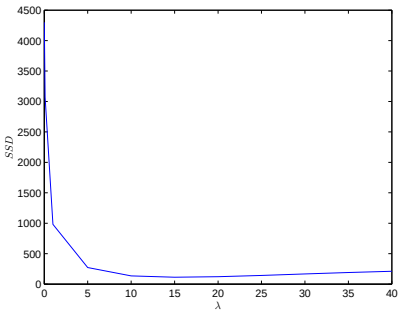

Fig. 6: Plot of SSD vs. λ. Minima of $SSD = 112.75$ for $\lambda = 15$.

estimate and the initial, noisy image. Moreover, we observed an increase in the number of iterations necessary to achieve convergence when λ is small. A very high λ corresponds to a weak regularization. In other words, we emphasize the importance of the correspondence between our estimate and the noisy image. When increasing λ, starting from a low value, we observe a sharp decrease in SSD with an optimum for intermediate values, followed by a gradual increase in error for large values.

When we compare algorithm 1 (gradient descent) with algorithm 2 (primal-dual method), we observe that regarding the SSD both methods reach a comparable optimal value. Furthermore, the behavior of the SSD and convergence with respect to λ is for both algorithms the same (figure 4 and 6). Consequently, we can state that both algorithms reach an equivalent solution. However, if we compare the algorithms with respect to computational time, we can see a distinct advantage for the primal-dual method, which converges in 0.75 seconds (averaged over 5 runs) compared to 31.87 seconds for the gradient descent ($\lambda = 15$).

In summary, the regularization parameter λ has a strong influence on the final result and should be carefully chosen (i.e. optimized). The primal-dual algorithm shows superior computation time compared to the gradient descent method.

Appendices

A Derivation of $\nabla_u E(u)$

$$
\begin{aligned}
E(u) &= \frac{\lambda}{2} LSQ(u,g) + TV(u) \\
\Rightarrow (\nabla_u E(u))_{i,j} &= \frac{\partial LSQ(u,g)}{\partial u[i,j]} + \frac{\partial TV(u)}{\partial u[i,j]} \\
\frac{\partial LSQ(u,g)}{\partial u[i,j]} &= \lambda(u[i,j] - g[i,j]) \\
\frac{\partial TV(u)}{\partial u[i,j]} &= \frac{\partial \tau[i,j]}{\partial u[i,j]} + \frac{\partial \tau[i-1,j]}{\partial u[i,j]} + \frac{\partial \tau[i,j-1]}{\partial u[i,j]} \\
\frac{\partial \tau[i,j]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]} \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2} \\
&= \frac{2u[i,j] - u[i+1,j] - u[i,j+1]}{\tau[i,j]} \\
\frac{\partial \tau[i-1,j]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]} \sqrt{(u[i,j] - u[i-1,j])^2 + (u[i-1,j+1] - u[i-1,j])^2} \\
&= \frac{u[i,j] - u[i-1,j]}{\tau[i-1,j]} \\
\frac{\partial \tau[i,j-1]}{\partial u[i,j]} &= \frac{\partial}{\partial u[i,j]} \sqrt{(u[i+1,j-1] - u[i,j-1])^2 + (u[i,j] - u[i,j-1])^2} \\
&= \frac{u[i,j] - u[i,j-1]}{\tau[i,j-1]} \\
\Rightarrow (\nabla_u E(u))_{i,j} &= \lambda(u[i,j] - g[i,j]) + \frac{2u[i,j] - u[i+1,j] - u[i,j+1]}{\tau[i,j]} \\
&\quad + \frac{u[i,j] - u[i-1,j]}{\tau[i-1,j]} + \frac{u[i,j] - u[i,j-1]}{\tau[i,j-1]}
\end{aligned}
$$

References

[1] P. Cattin. Lecture Notes in "Introduction to Signal- and Image Processing", 2013.

[2] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex

problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1), 2010.

[3] Q. Chen, P. Montesinos, Q. S. Sun, P. A. Heng, and D. S. Xia. Adaptive total variation denoising based on difference curvature. *Image and Vision Computing*, 28(3):298–306, Mar. 2010.

[4] S. Diego. Minimization of Functions Having Lipschitz Continous First Partial Derivatives. *Pacific Journal of Mathematics*, 16(1):2–5, 1966.

[5] N. Parikh and S. Boyd. Proximal Algorithms. 1(3):123–231, 2013.