ROF model for image denoising

1 An energy for edge-preserving image denoising

We use the same notation as in the previous assignments. We consider a scalar image $f : \Omega \to \mathbb{R}$ defined over the rectangular discrete lattice $\Omega = \{1, \dots, M\} \times \{1, \dots, N\}$ (N columns and M rows). Assume f has some noise we want to remove.

In Lab 2, we performed image denoising by minimizing an energy similar to the following quadratic energy:

$$\begin{aligned} E_2(u) &= \sum_{i=1}^{M}\sum_{j=1}^{N}((\nabla_i^+ u_{ij})^2 + (\nabla_j^+ u_{ij})^2) + \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{N}(u_{ij} - f_{ij})^2, \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N}|\nabla^+ u_{ij}|^2 + \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{N}(u_{ij} - f_{ij})^2. \end{aligned}$$

As we observed, this energy has a problem: it smoothes the edges of the image. The ROF denoising model is a slight modification of this energy:

$$\begin{aligned} E(u) &= \sum_{i=1}^{M}\sum_{j=1}^{N}\sqrt{(\nabla_i^+ u_{ij})^2 + (\nabla_j^+ u_{ij})^2} + \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{N}(u_{ij} - f_{ij})^2 \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N}|\nabla^+ u_{ij}| + \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{N}(u_{ij} - f_{ij})^2. \end{aligned}$$

We will see, that this slight modification has important consequences.

Removing the non-differentiability with an auxiliary variable

The energy E is not differentiable. Indeed, it has as many non-differentiabilities as pixels: one for each term $|\nabla^+ u_{ij}|$. For each one of them we add an auxiliary variable $p_{ij} \in \mathbb{R}^2$ and express the energy as a min-max problem:

$$\min E(u) = \min_u \max_{p \in C} \sum_{i=1}^{M} \sum_{j=1}^{N} \nabla^+ u_{ij} \cdot p_{ij} + \frac{1}{2\lambda} \sum_{i=1}^{M} \sum_{j=1}^{N} (u_{ij} - f_{ij})^2$$

Note that we have arranged all the dual variables in a vector-valued image $p \in \mathcal{Y}$. The set C is given by

$$C = \{p : \Omega \to \mathbb{R}^2 : |p_{ij}| \leq 1\}$$

It is the set of all vector valued images which in all pixels ij have a vector p_{ij} of norm smaller or equal than one. (Keep in mind that $p_{ij} \in \mathbb{R}^2$ and that $|\cdot|$ is the norm in $\mathbb{R}^2$.)

Now we can use the vector representation of images and note that we can express the energy as follows

$$\min E(u) = \min_u \max_{p \in C} \langle \nabla^+ u, p \rangle_{\mathcal{Y}} + \frac{1}{2\lambda} \|u - f\|^2_{\mathcal{X}}$$

This way of writing the energy will facilitate the computation derivatives. Let us define the primal-dual energy

$$G(u, p) = \langle \nabla^+ u, p \rangle_{\mathcal{Y}} + \frac{1}{2\lambda} \|u - f\|^2_{\mathcal{X}}$$

Note that this energy has the exact same form of the energies we studied in the previous lab. The only thing that changed is the form of the feasible set C.

This means that all that we did before applies to this case: the energy is convex with respect to u and concave with respect to p and the max-min problem is equivalent to the min-max.

We will solve the min-max problem using primal-dual and dual schemes exactly like we did before, with $A = \nabla^+, A^T = -\text{div}^-$, and $b = f$. We only need to modify P_C , the projector over C. For any vector-valued image $g : \Omega \to \mathbb{R}^2$, let $h = P_C(g)$ be its projection, then

$$h_{ij} = \frac{g_{ij}}{\max\{1, |g_{ij}|\}} = \frac{1}{\max\{1, \sqrt{g_{1,ij}^2 + g_{2,ij}^2}\}} (g_{1,ij}, g_{2,ij})$$

Observe that $|h_{ij}| \leq 1$ for all $(i, j) \in \Omega$ and therefore $h \in C$.

Primal-dual problem

In a primal-dual scheme we look for a saddle point (u^*, p^*) of G. A saddle-point is a minimum in u and a maximum in p. To find it we do an iterative scheme, updating u using a gradient descent and p with a gradient ascent.
Using the formulas derived in the previous part we have that:

$$\nabla_p G(u, p) = \nabla^+ u,$$

$$\nabla_u G(u,p) = -div^- p + \frac{1}{\lambda}(u - f).$$

Given an initialization u^0, p^0 , and time step parameters δ, $\theta > 0$, the primal-dual scheme is as follows:

$$u^{k+1} = u^k - \theta(\frac{1}{\lambda}(u^k - f) - \mathrm{div}^- p^k)$$

$$p^{k+1} = P_C(p^k + \delta\nabla^+ u^{k+1})$$

Note that since p is constraint to the set C, we have to use a projected gradient ascent.

1. *Complete the Python functions* `rof_primal_dual.m`. *Follow the comments provided in the code.*

Dual problem

The objective again is to solve the max-min problem:

$$\max_{p\in C} \min_u G(u,p) = \langle \nabla^+ u, p\rangle_{\mathcal{Y}} + \frac{1}{2\lambda}\|u - f\|^2_{\mathcal{X}}$$

The interior minimization problem with respect to u is easy to solve: it is a quadratic unconstrained problem. Thus, we can compute the minimum $u^*(p)$ as a function of p. From the previous part we get that

$$u^*(p) = f + \lambda \mathrm{div}^- p$$

Now we compute the dual energy as $E_D(p) = \min_u G(u,p) = G(u^*(p), p)$. The dual scheme is a projected gradient ascent on the dual energy E_D . Again, from the previous part we have that

$$E_D(p) = \langle \nabla^+ f, p\rangle_{\mathcal{Y}} - \frac{\lambda}{2}\|\mathrm{div}^- p\|^2_{\mathcal{X}}$$

$$\nabla E_D(p) = \nabla^+ f + \nabla^+ \mathrm{div}^- p = \nabla^+(f + \lambda \mathrm{div}^- p) = \nabla^+ u^*(p).$$

Putting all together, we have the following projected gradient ascent scheme to maximize the dual energy. The gradient ascent needs an initialization (any $p^0 \in C$, for example $p^0 = 0$) and a time step δ. Iterate the following:

$$p^{k+1} = P_C(p^k + \delta\nabla^+(f + \lambda \mathrm{div}^-(p^k))$$

The projection operator is the same as before. Note that the dual scheme does not need the computation of the primal variable. It only works on the dual domain. Once the dual maximization finishes, we obtain p^*, we compute the primal optimum as $u^*(p^*)$. In practice we will use the following scheme, which is exactly equivalent to the previous one, with the only difference that we identify the primal variable explicitly:

$$u^*(p^k) = f + \lambda \mathrm{div}^- p^k$$

$$p^{k+1} = P_C(p^k + \delta\nabla^+ u^*(p^k))$$

The only reason in doing so is because we are going to visualize the evolution of the image during the iterations. In real applications, this is not necessary.

2. *Complete the Python functions* `rof_dual.m`. *Follow the comments provided in the code.*
3. *Execute the Python script* `rof_main.m`. *Try different values of* λ. *Answer (briefly) this questions:*

1. *Which is the effect of* λ *on the results?*
2. *Are the edges of the image well preserved? What happens with the textures?*
3. *Look at graphs of the evolution of the dual gap with the iterations. What do you observe? What happens when you change step size* δ*?*
4. *Use the algorithm with new examples, for example, own photos.*