Variational Methods for Computer Vision: Exercise Sheet 7

Exercise: December 12, 2017

Modified version (December 11, 2017) of Part I, Exercise 1.

Part I: Theory

The following exercises should be **solved at home**. You do not have to hand in your solutions, however, writing it down will help you present your answer during the tutorials.

1. Let $Q := [0,1] \times [0,1]$ be a rectangular area and let $V = (u(x,y), v(x,y))$ be a differentiable vector field defined on Q.

   (a) Prove Green's theorem for Q using the fundamental theorem of calculus, hence show that

   $$\iint_Q \operatorname{curl} V \mathrm{dxdy} = \iint_Q \big(v_x(x,y) - u_y(x,y)\big)\mathrm{dxdy} = \oint_{\partial Q} V(s) d\vec{s}.$$

   Assume the boundary curve ∂Q to be oriented counterclockwise.

   (b) Let $\Omega \subset \mathbb{R}^2$ be an area that can be represented as a disjoint union of a finite number of squares $Q_1, ..., Q_n$. Prove that Green's theorem also holds for Ω.

2. In the lecture the piecewise constant Mumford-Shah functional is written as follows:

   $$E(u_i, C) = \sum_{i=1}^{n} \int_{\Omega_i} (I(x) - u_i)^2 \, dx + \nu |C|.$$

   Prove that by merging two regions Ω_1 and Ω_2 the energy E changes by

   $$\delta E = \frac{A_1 A_2}{A_1 + A_2} (u_1 - u_2)^2 - \nu \delta C,$$

   where A_i denotes the area of the regions in pixels, u_i the respective mean values and δC the length of the interface of both regions.

Part II: Practical Exercises

This exercise is to be solved **during the tutorial**.

1. Finish the practical exercises from the last exercise sheet.

   The following hints address some of the questions that were raised during last week's tutorial.

   - One reason why we implement the linear operators as matrices, rather than for example using Matlab's builtin functions to perform Gaussian blurring, is that with matrix representations it is easy to compute the adjoint, which is simply the transpose of those matrices.
   - You can implement explicit or implicit time discretization for the gradient descent. When you implement explicit time discretization, you can speed things up by making sure the during computation of the descent direction, you group operations such that you only compute matrix-vector products, not matrix-matrix products. If you do implicit time discretization, you need to explicitly construct a (sparse) matrix in each step. You should pre-compute all matrix-matrix multiplications that are idenpendent from the current value of the optimized image before the optimization loop to safe computation time.
   - For the spatial gradient, the Gaussian smoothing, and the shifting (to fill up the missing values), you may implement von Neumann boundary conditions. To keep things simple, you can initially ignore the boundary conditions, if you want to. The result in the interior of the image should not be affected much.
   - You may want to consider also down-scaling factors larger than 2 (e.g. 4, 8, ...) to see the super-resolution effect more clearly.
   - To compute the downsampled observations f_i, the exercise sheet suggests to shift, blur, and then `imresize` with nearest neighbor interpolation. This is fine, but in particular for larger down-scaling factors, the nearest neighbor interpolation does not model the behaviour of a real camera well. You should add the block-wise averaging before downsampling, i.e. in total: Shift, blur, block-wise average, downsample. Block-wise-averaging, down-sampling, and then upsampling is equivalent to just the averaging. I.e. you may skip implementing the upsampling operator and instead optimize

     $$E(u) = \sum_{i=1}^{n} \int_\Omega ((ABS_i u)(x) - f_i(x))^2 \,\mathrm{dx} + \lambda \int_\Omega |\nabla u(x)| \,\mathrm{dx}, \tag{1}$$

     where $f_i = ABS_i f$.