Variational Methods for Computer Vision: Exercise Sheet 6

Exercise: 09 December 2015

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 $V = (v(x,y), u(x,y))$ be a differentiable vector field defined on Q.

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

   $$\int_Q v_x(x,y) - u_y(x,y)\ dxdy = \oint_{\partial Q} V(s) d\vec{s}$$

   Assume the boundary curve ∂Q to be be oriented counter clockwise.

   (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 the Green 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.

3. Let $\Omega = [-5; 5] \times [-5; 5]$ be a rectangular area and let $I : \Omega \to [0; 1]$ be an image given by :

   $$I(x,y) = \begin{cases} 1 & \text{if} \quad x^2 + y^2 \leq 1 \\ 0 & \text{else} \end{cases}$$

   Furthermore let $C : [0,1] \to \Omega$ be a curve represented by a circle centered at the origin having radius r.

   (a) Write down the Gateaux-Derivatives of the Mumford Shah functional for 2 regions for the two cases $r > 1$ and $r \leq 1$.

   (b) Show that the Gateau derivative at $r = 0$ is not continuous. Why is $\nu \leq 1$ a good choice in order to obtain good segmentation results. What is the ideal choice for ν in our example?

Part II: Practical Exercises

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

Super-Resolution from Video.
In the lecture we encountered the concept of super resolution from video. The key idea of super resolution is to exploit redundancy available in multiple frames of a video. Assuming that each input frame is a blurred and downsampled version of a higher resolved image u, the high-resolution image can be recovered as the minimum of the following energy functional:

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

The Linear Operator B denotes a Gaussian Blurring. The upsampling operator U simply replaces every pixel with four pixels of the same intensity. In order to be able to compare image u with the upsampled version of f_i which is constant blockwise, we apply the linear averaging operator A on u which assigns every block of pixels the mean values of the pixels in that block. The linear operator S_i accounts for the coordinate shift by motion s_i hence:

$$(S_i u)(x) = u(x + s_i(x)).$$

1. In the following we are going to construct a toy example for super resolution by executing the following steps:

   (a) Download the archive `vmcv_ex07.zip` and unzip it on your home folder. In there should be a file named `Boat.png`.

   (b) Create from the unzipped image 6 versions shifted in x direction by exactly one pixel hence:
   $$f_i(x, y) = f(x + i, y),$$
   for $i = 1...6$. In order to account for the boundary, consider taking cropped images from the interior of the original image.

   (c) In order to simulate blurring convolve the shifted images with a gaussian kernel. Next downsample the images f_i by factor 2 by using the `imresize` function in Matlab with nearest neighbor interpolation.

2. In what follows we are going to minimize the above functional in order to obtain a super resolved image from our input images f_i.

   (a) Derive the Euler-Lagrange equation of E and the corresponding gradient descent scheme.

   (b) Compute the matrix representations of the linear operators A, B, S_i and U. Since these matrices are huge, again use sparse data structures in Matlab (`spdiags speye`) in order to obtain a sparse representation.

   (c) Compute $u^* = \mathrm{argmin}_u\ E(u)$ by means of gradient descent using matrix vector representation after stacking the function u in a vector using the matlab command `reshape`.