Weekly Exercise 8

Dr. Csaba Domokos
Technische Universität München, Computer Vision Group
June 26th, 2017

Programming (7+7 Points)

Exercise 1 (Semantic image segmentation with FastPD1, 7 Points). Consider the *energy function* for $w \geq 0$

$$E(\mathbf{y};\mathbf{x}) = \sum_{i \in \mathcal{V}} E_i(y_i; x_i) + w \sum_{i,j \in \mathcal{E}} E_{ij}(y_i, y_j; x_i, x_j) \quad , \tag{1}$$

for the *multi-labeling problem*, i.e. $\mathbf{y} \in \mathcal{L}^{\mathcal{V}}$, where $\mathcal{L}$ stands for the label set. $\mathcal{V}$ is the set of pixels of $\mathbf{x}$ and $\mathcal{E}$ consists of all four-neighboring pixels. Implement the *FastPD1 algorithm* to solve **semantic image segmentation** for the images shown in figure 1. Try to choose different values for the parameter w for Equation (1) and compare the segmentation results.

Figure 1: The test images for semantic image segmentation.

These test images have been obtained from the MSRC image understanding dataset[1], which contains 21 classes, i.e. $\mathcal{L} = \{1, 2, \ldots, 21\}$. The meaning of the classes are given in the `21class.txt` file. Use it to check whether your results are reasonable.

To define the **unary energy functions** E_i, use the `*.c_unary` files provided in the supplementary material (`supp_07`). Each test image has its own unary file, specified by the same filename. From each unary file, you can read out a $K \times H \times W$ array of float numbers. The H and W are the image height and width, and $K = 21$ is the number of classes. This array contains the 21-class probability distribution for each pixel. You may find the `multilabel_demo.cpp` in the supplementary material, which demonstrates how to load a unary file and read out the corresponding probability values. The unary energy functions E_i for all $i \in \mathcal{V}$ are then defined as the *negative log-likelihood*.

[1] https://www.microsoft.com/en-us/research/project/image-understanding/

The **pairwise energy functions** E_{ij} are defined by the *contrast sensitive Potts-model*

$$E_{ij}(y_i, y_j; x_i, x_j) = \exp(-\lambda \|x_i - x_j\|^2)[\![y_i \neq y_j]\!] \ ,$$

where x_i is the intensity vector for pixel i, and you may choose $\lambda = 0.5$.

Exercise 2 (Branch–and–MinCut[2], 7 Points). In this exercise we apply the *branch and bound method* to find a global minimizer of a discrete version of the celebrated Chan–Vese[3] segmentation energy function:

$$E(\mathbf{y}, \{c_f, c_b\}) = \sum_{i \in \mathcal{V}} \left(\nu + \lambda_1 (I_i - c_f)^2\right) y_i + \sum_{i \in \mathcal{V}} \lambda_2 (I_i - c_b)^2 (1 - y_i) + \mu \sum_{(i,j) \in \mathcal{E}} |y_i - y_j| \ . \quad (2)$$

Here I denotes a gray-scale input image with $|\mathcal{V}|$ pixels, i.e. at every pixel $i \in \mathcal{V}$ we have $I_i \in [0, 255] \cap \mathbb{N}$. The variable $\omega = (c_f, c_b) \in \Omega = [0, 255]^2 \cap \mathbb{N}^2$ denotes the mean intensity of the foreground respectively the background for the segmentation $\mathbf{y} \in \mathbb{B}^n$.

Compute a global minimizer of (2) using the *branch and bound best–first* tree search. The search space Ω is the rectangle $[0, 255]^2 \cap \mathbb{N}^2$. In your implementation, you can keep a priority queue of rectangles Ω_t. In every iteration you may remove the rectangle with the smallest lower bound (for more details please refer to the lecture) and split it into two smaller rectangles along the longest edge.

Figure 2: The figure shows the input image and a global minimizer of (2) for parameters $\lambda_1 = \lambda_2 = 0.0001$, $\mu = 1$ and $\nu = 0.1$. The optimal foreground and background colors were found as $c_f^* = 81$ and $c_b^* = 167$.

[2] V. Lempitsky, A. Blake, C. Rother, Image Segmentation by Branch–and–MinCut, ECCV, 2008

[3] T. Chan, L. Vese: Active contours without edges. IEEE Trans. on Image Processing, 10(2), 2001.