Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

universitätbonn

Practical Lab
Variational Methods and Inverse Problems in Imaging

Summer term 2014

Prof. Dr. M. Rumpf – A. Effland, B. Geihe, S. Simon, S. Tölkes

Problem sheet 2 **May 16th, 2014**

Problem 2 (Chan-Vese segmentation)

During the next weeks we will study the Chan-Vese level set approach for the piecewise constant Mumford Shah model. To obtain a segmentation the following energy is to be minimized:

$$E_{\text{CV}}[c_1, c_2, \phi] = \frac{1}{2} \int_\Omega (u - c_1)^2 (1 - H(\phi)) + (u - c_2)^2 H(\phi)\, \mathrm{d}x + \nu \int_\Omega |\nabla H(\phi)|_\epsilon \, \mathrm{d}x \,.$$

Our ultimate goal is to implement the alternating minimization algorithm discussed in the lecture. To lay the foundations we will set up all required finite element operators to be able to assemble and solve the system of equations for one semi implicit time step:

$$\left(M[\Phi^k] + \tau L[\Phi^k] \right) \bar{\Phi}^{k+1} = \tau \bar{F}[c_1, c_2] + M[\Phi^k] \bar{\Phi}^k \,, \quad \text{where}$$

$$M[\Phi] := \left(\int_\Omega \mathcal{I}_1(\psi_i \psi_j) \mathcal{I}_0 \left(H'_\rho(\Phi)^{-1} \right) \, \mathrm{d}x \right)_{i,j \in I}$$

$$L[\Phi] := \left(\nu \int_\Omega \frac{\mathcal{I}_0(\nabla \psi_i \cdot \nabla \psi_j)}{\mathcal{I}_0(|\nabla \Phi|_\epsilon)} \, \mathrm{d}x \right)_{i,j \in I}$$

$$\bar{F}[c_1, c_2] := \left(\frac{1}{2} \int_\Omega \mathcal{I}_0 \left(((c_1 - u_0)^2 - (c_2 - u_0)^2) \psi_i \right) \, \mathrm{d}x \right)_{i \in I}$$

We approximate the Heaviside function by

$$H_\rho(s) = \frac{1}{2} \left(1 + \frac{2}{\pi} \arctan \left(\frac{s}{\rho} \right) \right) \,.$$

For fixed Φ^k we can compute the optimal intensities c_1^k and c_2^k by

$$c_1^k = \frac{\int_\Omega \mathcal{I}_0 \left(u_0 (1 - H_\rho(\Phi^k)) \right) \, \mathrm{d}x}{\int_\Omega \mathcal{I}_0 (1 - H_\rho(\Phi^k)) \, \mathrm{d}x}, \quad c_2^k = \frac{\int_\Omega \mathcal{I}_0 \left(u_0 H_\rho(\Phi^k) \right) \, \mathrm{d}x}{\int_\Omega \mathcal{I}_0 \left(H_\rho(\Phi^k) \right) \, \mathrm{d}x} \,.$$

Your tasks (May 9th, 2014):

(i) Write a class MyHeavisideFunction with member functions evaluate and evaluateDerivative, where ρ is passed to the constructor.

(ii) Derive classes from FENonlinIntegrationScalarInterface to compute the numerators and the denominators of c_1^k and c_2^k. To evaluate u_0 at a given quadrature point you can use DiscreteFunctionDefault which then needs to be passed to the constructor of your operator. The piecewise constant interpolation $\mathcal{I}_0$ will be achieved by using an appropriate quadrature rule, i. e. a quadrature of order 1 which is evaluated at the center of mass.

(iii) Implement the weighted mass matrix by deriving from LumpedMassOpInterface and implementing getCoeff. This interface will exploit the fact that mass lumping leads to diagonal matrices. Pass the nodal values vector of the level set function $\bar{\Phi}$ as a DiscreteFunctionDefault object. To evaluate this discrete function in the center of mass either use evaluate with local coordinates $(0.5, 0.5)$ or set the quadrature rule for the DiscreteFunctionDefault object to order 1 and use evaluateAtQuadPoint.

(iv) Likewise implement the weighted stiffness matrix by overwriting getCoeff of FELinScalarWeightedStiffInterface. Again, pass $\bar{\Phi}$ as a DiscreteFunctionDefault object and choose the quadrature rule appropriately. Recall $|x|_\epsilon = \sqrt{x^2 + \epsilon^2}$.

(v) For the vector $\bar{F}$ derive from FENonlinOpInterface and implement getNonlinearity. Again, choose the quadrature rule appropriately and pass u_0.

(vi) Finally try to compute c_1 and c_2 for fixed u_0 and Φ and solve the system of equations for fixed τ. This will later become a single step of the alternating minimization algorithm.

Your tasks (May 16th, 2014):

Implement the full Chan-Vese model with $\tau_0 = 512$, $\rho = 10^{-4}$, $\nu = 5 \cdot 10^{-3}$ and $\epsilon = 10^{-5}$!

Some hints:

(i) The following headers should be included:

adaptiveGridPlotter.h, adaptiveQuocUtils.h

(ii) To load an image, you can either use the constructor of a ScalarArray or the method loadPNG. Afterwards you have to run the AdaptiveArrayConverter for compatibility issues with the adaptive grids. Furthermore, the AdaptiveGridPlotter allows you to save your results.

(iii) We provide the class SemiImplicitGradientDescent to perform the semi-implicit gradient descent. Derive a class ImplicitPart from

aol :: SemiImplicitGradientDescentImplicitPart

and overload the following methods:

assembleSystemMatrix, constructRHS, setTau.