Assignment 3: Segmentation: Snakes. Vision and Image Processing

Kim Steenstrup Pedersen, François Lauze, and Jan Kremer

December 16, 2013

This is the third mandatory assignment on the course Vision and Image Processing. The goal is for you to get familiar with programming for computer vision with a on classical segmentation algorithms.

You have to pass this and the following mandatory assignments in order to pass this course. There are in total 4 mandatory pass/fail assignments. This is a **group assignment**, i.e., we expect that you will form small groups of 2 to 4 students that will work on this assignment.

The deadline for this assignment is Wednesday 8/01 2014. You must submit your solution electronically via the Absalon home page. Go to the assignments list and choose this assignment and upload your solution prior to the deadline. Remember to include your name and KU user name (login for KUnet) in the solution. If you do not pass the assignment, having made a SERIOUS attempt, you will get a second chance of submitting a new solution.

A solution consist of:

- Your solution source code (Matlab / Python scripts / C / C++ code) with comments about the major steps involved in each Question (see below).
- Your code should be structured such that there is one main file that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions / classes.
- Your code should also include a README text file describing how to compile (if relevant) and run your program, as well as list of all relevant libraries needed for compiling or using your code. If we cannot make your code run we will consider your submission incomplete and you may be asked to resubmit.
- The code, auxiliary files and README file should be put into a compressed archive file in either the ZIP or tar format (RAR is not allowed - we simply cannot read your archive).
- A PDF file with notes detailing your answers to the questions, which may include graphs and tables if needed (**Max 5 pages** text including figures and tables). Do NOT include your source code in this PDF file.

A note on relevant software

We recommend that you select either Matlab / Python / C / C++ as the programming language you use for your solutions for these assignments. We

also recommend that you select the language you are most confident in - the focus should be on learning the methods and not learning a new programming language.

If you wish to use Matlab, the University of Copenhagen has a license agreement with MathWorks that allow students to download and install a copy of Matlab on personal computers. On the KUnet web site you find a menu item called Software Library (Softwarebiblioteket):
`https://intranet.ku.dk/Sider/Software-bibliotek.aspx`.
Under this menu item you can find a link to The Mathworks - Matlab & Simulink + toolboxes. Click this link and follow the instructions for how to install on your own computer.

If you wish to program your solutions in C++ we recommend the use of the CImg Library, which can be obtained at `http://cimg.sourceforge.net`

Snakes

The goal of this assignment is to implement the classical algorithm of M. Kass, A. Witkin and D. Terzopoulos. A copy of the original paper is available for download in Absalon.You should start testing with very simple images, like a white rectangle on a black background and gradually more complex ones.

You will have to test a couple of external energy terms:

- Going to the local maxima of the gradient magnitude squared of the image at scale σ:
$$E_1(C) = -\frac{1}{2}\int_0^N \|g_\sigma \star \nabla I(C(t))\|^2 \, dt$$
- Going to the zero-crossing of the Laplacian of Gaussian of the image at scale σ (i.e. incorporating the Marr-Hildreth edge detector):
$$E_2(C) = \int_0^N -\left(g_\sigma \star \Delta I(C(t))\right)^2 \, dt$$

g_σ is the 2D-Gaussian of standard deviation σ.

The implementation should solve a number of different tasks, that will be listed in the following sections.

1 Getting the initial curve

You will write a procedure that allows you to specify a curve from a series of points selected in the image in a more or less interactive way. Both Matlab and Python (matplotlib) provide a routine `ginput` that you could use. In general you may select a relatively small number of points and construct a curve with added points using a form of 1-dimensional interpolation for both the x and y coordinates. Matlab provides an `interp1d` routine that will be helpful. Python provides a similar mechanism in the module `scipy.interpolate`

2 Gradient of the external energy

Compute formally the gradients of the above mentioned external energies. Then implement a procedure that gives the corresponding contributions of these energy gradients along a curve (which will be our evolving snake). Not that for Python users, I provide a simple interpolation module that will be useful for that purpose.

3 The system matrix

The System matrix is the inverse of the pentadiagonal matrix that encodes the different curve differentiation operations with the corresponding α, β weights as well as the evolution time step τ. This is a square matrix of size given by the number of points. Both Matlab and Python have the necessary routines to inverse the pentadiagonal matrix and provide the system matrix. Write a procedure that build it.

4 Putting everything together

You will have to assemble the different elements in a program that will run the Snake simulation. You are encouraged to develop an interactive program that will help you understanding the different parameters.

5 Testing

You should first run you algorithm with external energy part. This should demonstrate the curve smoothing regularizing properties of the internal energy minimization. It is enough to set the external energy weight to 0 to do so.

Then you will apply it to a couple of images and the above mentioned energies. Notice that the curve regularization weights should in general be must smaller than the external energy one.