Assignment 1: Filtering and edge detection
Vision and Image Processing

Søren Olsen, Francois Lauze and Hong Pan

November 8, 2015

This is the first mandatory assignment on the course Vision and Image Processing. The goal for you is to get familiar with computer vision programming with a focus on filtering. Filtering is a core discipline in image processing and forms the basis for feature extraction which again is central to many of the techniques we are going to study.

This assignment must be solved individually. You have to pass this and the following 3 mandatory assignments in order to pass this course. If you do not pass the assignment, but you have made a SERIOUS attempt, you will get a second chance of submitting a new solution.

The deadline for this assignment is Wednesday 2/12, 2015 at 20:00. You must submit your solution electronically via the Absalon homepage. 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.

Filtering and edge detection

Filtering (really Finite Input Responce filtering) is done by convolutions. Among the most applied filters are the Gaussian and its first and second order derivatives. Convolution with a Gaussian itself will blurr the image. Convolutions with the two first order Gaussian derivatives provides an estimate of the gradient field from which the gradient magnitude may be derived and visualized. Convolution with the sum of the second order (unmixed) partial derivatives of a Gaussian (dubbed The mexican hat operator) may be used both to detect blobs (as the local extremes) and edges (as the zero crossings). Edges and blobs in images code much of the semantic information available in the images and are often used as building elements in further analysis.

In this assignment you must demonstrate that you can write programs for image filtering and edge detection. Please check the note below on the use of relevant software. In detail you must demonstrate that you and implement, perform and evaluate:

- Gaussian filtering. Show the result using $\sigma = 1, 2, 4, 8$ and explain in detail what can be seen.
- Gradient magnitude computation using Gaussian derivatives. Use $\sigma = 1, 2, 4, 8$, and explain in detail what can be seen and how the results differ.
- Mexican hat filtering. Again, use $\sigma = 1, 2, 4, 8$, and explain in detail what can be seen and how the results differ.
- Canny edge detection. Select what you think is a set of good parameter values, apply, show, and compare with your previous results.

You should apply the abovementioned methods to the image **lenna.jpg** available at Absalon. However, you are encouraged to use other images as well (in particular very simple images, that more easily let you verify your result). Please notice that for display purpose some filtering results probably need to be remapped into [0:1] or whatever your display routine requires.

Unless explicit stated otherwise, during this course you are allowed to apply routines available in any library that you may find useful. For this assignment in particular, you are not supposed to implement neither Canny edge detection nor Gaussian convolutions (although the latter might be a good programming exercise). Please see the note on relevant software below. Also, please recognize that the less you demonstrate your ability to write relevant (non-trivial) code, the more you are expected to conduct experiments and (in particular) to comment on these and to present your new knowledge gained by the experiments.

As described in detail below your answer should include your code and a pdf-file describing your solution and showing examples of well-chosen image results. Each image/graph or other illustration should be commented (don't trust that I see what you see). In detail a solution consists of a zip-file containing:

- A PDF file describing your answers to the assignment, which may include images, graphs and tables if needed. You are allowed to used **Max 8 pages** of text including figures and tables). We recommend to write only 2-3 pages of text excluding all graphics. Images should be shown large enough to show what you want me to see (if printet on A4 paper). Do NOT include your source code in this PDF file.
- Your solution source code in original plain text format (Matlab / Python scripts / C / C++ code) with comments about the major steps involved in each question.
- 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.
- A README text file describing how to compile (if relevant) and run your program, as well as a 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.

Please notice once again, that all documents and all code should be put into a single compressed archive file in either the ZIP or TAR format (RAR is not allowed - we simply cannot read your archive).

A note on relevant software

We recommend that you select either Matlab, Python, C, or C++ as the programming language you use for your solutions for the assignments on this course. 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 allows 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/selvbetjening/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 use C / C++ we recommend that you use the OpenCV library `http://opencv.org/` and/or the VLFeat library `http://www.vlfeat.org/`. Both libraries provide an extensive collection of implementations of central computer vision algorithms. OpenCV also provides an interface for Python. Follow the installation instructions on these websites to install on your own computer. OpenCV is also available via MacPorts on MacOSX.