COMP 558 Assignment 1

Prepared by Prof. Michael Langer

Posted: Fri. Sept 21, 2018
Due: Fri. Oct. 5, 2018 (midnight)

Introduction

This assignment covers the material from lectures 2 and 3. The questions concern basic image processing using convolution, and the core ideas of the edge detection methods of Canny and Marr-Hildreth.

In order to do the assignment, you need to know how to use Matlab and be used to indexing conventions for matrices and plots and the various image processing commands. If you do not yet know Matlab, then learning these tools will take some time and will be part of the work that you do in the assignment. Always look at Matlab's documentation for help. If you go on the Mathworks website, you can also find webinars and tutorials for various computer vision/ image processing tasks.

Please use MyCourses for discussion and ask if you have doubts. The instructors and TAs will be active there. We also encourage you to discuss the questions with each other and to give each other hints and tips – not just about Matlab related issues, but also about the specific assignment questions. *However, the solutions that you submit must be your own work.* You are not permitted to copy code from each other or from the internet.

Instructions

Submit a single zipped file **A1.zip** to mycourses Assignment 1 folder. The zip file must contain:

- a PDF with figures and text for each question (We suggest that you do not spend time with fancy typesetting. Please just make sure the explanations are clear and the figures are easy to read. No points for visual beautiful in this course!)
- the Matlab code that you wrote
- the images that you used.

In order to receive full points, your solution code must be properly commented, and you must provide a clear and concise description of your computed results in the PDF. Sbe concise and clear! Note that the TA's have limited time resources and will spend at most 30 minutes grading each assignment.

Late assignment policy: Late assignments will be accepted up to only 3 days late and will be penalized by 10 points per day, e.g. An 80 grade would be reduced to 72 for one day late. If you submit only a few minutes late, we reserve the right to treat this as "one day late".

Question 1: Filtering (30 points)

a) Implement a Matlab function that returns a 2D Gaussian matrix which can be used for filtering an image. Here is the template for this function which you must use:

```
function g = make2DGaussian(N, sigma)
%  N is odd, and so the origin (0,0) is positioned at indices
%   (M+1,M+1) where N = 2*M + 1.
end
```

Doing this question ensures that you understand what a 2D Gaussian is, and how matrix indices are defined in Matlab. In particular, the definition of convolution in Matlab is slightly different from the one given in class. https://www.mathworks.com/help/matlab/ref/conv.html The reason for the difference has to do with Matlab indices starting at 1 rather than 0.
Remember: In the tutorial, Pulkit went over how you might fill in the values in a filter matrix.

b) Implement a Matlab function that performs a 2D convolution, similar to the Matlab builtin function `conv2`. Here is the template for this function which you must use:

```
function im =  myConv2(image, filter)
end
```

Note the order of the arguments. It was mentioned in the lectures that convolution is commutative. However, when one filters an image, one needs to choose how to handle pixels at the image boundary and to decide how big the resulting filtered image should be. For your implementation, we require that you treat the original image as zero padded and the output must be the same size as the input image (the first argument).

Matlab provides a built-in function `fspecial` and `conv2` as we saw in the tutorial for filtering a gray level image with a Gaussian. We recommend that you use these functions to verify the correctness of your implementations above. For the rest of this assignment, we suggest you use these functions rather than your own implementations. This will also protect you from propagating errors, in case your implementations are incorrect.

You should submit the Matlab code but you do not need to submit an image for testing. We will test your two functions on our own.

Question 2: Edges detection using intensity gradients (20 points)

This question requires you to work with a real image. Use your cell phone or a digital camera to take an image of a scene that contains many objects, some of which have a long line structures. For example, the objects might be books, window frame, doorway, shelf, etc. Read the image into Matlab, and resize it with the builtin `imresize` function so that you can display it in full resolution. A typical iphone picture is 12 MP is too big to show on your laptop screen in full resolution, and which could take long to process anyhow. So, for example, you could resizse such a 3k x 4k image to 300x400.

Decompose the image into RGB channels. Take the green channel, make a gray scale JPG image of it, and include this in your submission. You will work with just that channel.

Filter the image with a Gaussian and take local differences (central as in the lectures) in the x and y directions. (This gives you the intensity gradient at each pixel.) *Show the pair of filtered images in your PDF.*

Compute the gradient magnitude and the gradient orientation at each pixel. The orientation is an angle spanning a range of 2π radians. Mark each pixel as an edge if its gradient magnitude exceeds some large threshold. Choose the threshold by hand so that the computed edges roughly correspond to the edges that you perceive when you look at the image. *Show the resulting binary image which indicates the edge positions.*

Repeat the filtering for two more values of σ that are 2x and 4x as big as what you used originally. *Show the resulting edge maps and discuss any differences in the results.* You don't need to show all previous images – just the edge maps.

Question 3: Edge detection using Marr-Hildreth and zero crossings (30 points)

Create an image that consists of several rectangles and disks at different positions and of different sizes and grey levels, i.e. not RGB. Some of the rectangles and disks should overlap each other.

Implement a Laplacian of a Gaussian filter defined by parameter σ and convolve it with your rectangle and disk image. Find the pixels at which there is a zero crossing in either the x or y direction, and create a binary image that indicates where these zero crossing points are. You can find zero-crossings by checking a 3x3 neighborhood, and finding a point where neighbors on the opposite side of a pixel have opposite signs of the filtered image. Again, repeat for three different σ values and give figures that show these zero-crossings (Marr-Hildreth edges).

Discuss how the positions of the zero-crossings depend on σ. Compare the behavior of the zero crossings for edges that are isolated versus edges that are close to "T junctions".

Hint: It may help you to browse through Marr and Hildreth's 1979 paper.

Question 4: Local comparison of gradients and edges (10 points)

From the images used in Questions 2 and 3, select a small region with say 20 x 20 pixels. Choose one of your σ values, and show a crop of that region in the image, and a crop of the edge maps. *Briefly discuss the relationship between these figures and plots.*

Use the Matlab `quiver` function to illustrate the image gradients, by overlaying the quiver plot on the original images in the selected region. *Compare how image gradients vary at the intensity edges, at regions of flat intensity etc.* Look at the Matlab documentation for help.

Question 5: frequency distribution of gradients (10 points)

The pixels of an image can be thought of as a matrix of random variables, such that the events are the image intensities. For real images, the intensities at neighboring pixels are not independent, since nearby pixels tend to receive light from nearby points in the 3D world and thus the intensities of neighboring points tend to be similar. Since nearby pixels tend to have similar intensities, the intensity *difference* of nearby pixels tends to be close to 0.

Use the Matlab builtin `imhist` function to plot a frequency distribution of the partial derivatives of Gaussian filtered (smoothed) images which you computed in Question 2.

The shape of the frequency distribution is superficially similar to a Gaussian. However, there is an important difference, namely the tails of the distribution are different from the tails of a Gaussian. To see this difference, it will help to plot the log of the frequency distribution, and consider the "tails" of the Gaussian. *Why does this difference occur, i.e. why is the distribution not Gaussian-like?*
(Note: A frequency distribution will integrate to the number of pixels, not to 1, but we are asking about the *shape* of the distribution here, so you can think of normalizing the frequencies by dividing by the number of pixels.)

Repeat the above histogram analysis for the different values of σ as in previous questions and discuss how the results change.

Get started early! Good luck, and have fun!