Computer Vision and Image Analysis
Assignment Sheet 1 - 21.10.2014

Next exercise group: 24.10.2014

- Exercise 1.1, first part, is a pre-requisite for all that follows - it is mandatory that you do it.
- Deadline for exercise: **28.10.2014, 13:30**, 30% point loss per day late.
  Please submit solutions via e-Mail in a single ZIP-Archive to `ole.johannsen@uni-konstanz.de`. See the course webpage for guidelines how to format the content, please follow them exactly to make his job easier.

Exercise 1.1 (mandatory)
The goal of the exercise is to get a working Matlab environment up and running and become familiar with the basics of handling images. The first few steps are just to give you tips on how to work with images in Matlab. You do not have to turn them in.

- Install Matlab. You can find a version online at
  `http://www.rz.uni-konstanz.de/en/our-offers/software/state-wide-matlab-agreement/` ,
  free for all students of Konstanz University.
- Check out some online tutorials:
  - Beginner's tutorials can be found at
    `http://www.mathworks.de/academia/student_center/tutorials/launchpad.html`
  - An overview of the image processing capabilities of Matlab can found at
    `http://www.mathworks.co.uk/help/images/examples.html`
    All examples on this page are already present in Matlab and can be opened using e.g.

    ```
    edit BlindImageDeblurringExample
    ```
  - A multitude of other tutorials can be found online.
- A list of commands you should be familiar with is *imread, imshow, imagesc, axis equal|image, colormap gray|hsv|jet....* Additionally you should familiarise yourself with pixelwise transforms. I.e. thresholding (e.g. *image<0.5* assigns a 1 to pixels with a value smaller than 0.5 and 0 otherwise) and pixelwise multiplication (*(image<0.5).*image* assignes a 0 to all pixels with values greater than 0.5 while keeping the other pixels unchanged). How do you add a specific value to each pixel?
- Another important part is selecting only specific parts of an image using *image(1:2:50,:)* or linearising an image to calculate statistics like *max(image(:))*.
- Please take a look at the debugging capabilities of Matlab. Breakpoints and the function of $F5, F10, F11$ are rather improtant. A useful script to close all figures, clear all variables and the command window, while keeping the breakpoints set can be found here:
  `http://stackoverflow.com/questions/12657219/how-to-restore-breakpoints-in-matlab-after-clear-all`.
  It might be useful to add this as *ccc.m* to /home/USERNAME/Documents/MATLAB (look at *path* in Matlab for your specific directory) so you can run it everywhere inside Matlab.

The remainder of the exercise should be handed in, you can get up 10 points in total if everything is correct.

(a) Create a Gaussian kernel with standard deviation σ, apply it to an image of your choice using *conv2* or *fspecial*. Visualize the kernel and the result.
Tip: `http://lmgtfy.com/?q=matlab+conv2` ;).

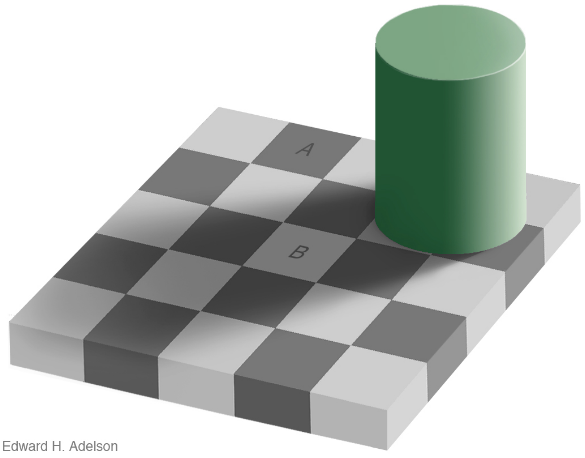
(b) Program your own convolution function without using *conv2* or similar functions. Use "clamp to border" as boundary conditions. Here two different paths should be taken. On the one hand a straightforward implementation using nested for-loops and on the other hand a version using matrix/vector operations.

(c) Compare your results to the builtin 'conv2'. Also compare the runtime of the approaches using *tic ... toc*. What does that tell you about the use of for-loops?

(d) Perform Gaussian convolutions with different σ and different kernel sizes, observe what happens.

(e) Visualize the difference of two Gaussian convolutions with two different standard deviations σ_1, σ_2. Vary the standard deviations a bit. How would you interpret the result?

Exercise 1.2
This exercise is mostly for fun and to develop some insights into perception. Remember the visual illusion from the lecture:



Square B appears visually brighter than A, although the absolute color is exactly the same. Of course, your brain is right in a sense: if this were an actual scene, square B would most likely be indeed have a brighter color, and the reason for its darker appearance is just that it lies in a shadow. The processing of the visual system which accounts for illumination effects on color is called "color constancy".

You can download the image here:

`https://depot.uni-konstanz.de/cgi-bin/exchange.pl?g=glebm7b6zg`

The illusion can be broken by e.g. turning all pixels except the ones in squares A and B white, but this seems like overkill - or is it not?

(a) Load the image in your favourite image editing application.

(b) Just to make sure, verify that the colors of A and B are indeed the same.

(c) Use the image editing tool to change the color of pixels. Try to "break" the illusion by changing as few pixels as possible to white - "breaking" in this case means that A and B again appear to have the same color for a human observer. Use Matlab to count the number N of pixels which you needed to change. The result with the smallest number will get a honorable mention in next years lecture.

(d) Now do it the other way around: Try to change as many pixels as possible to the color white while still keeping the illusion intact (i.e. A and B appear different to a human observer). This time, largest number M of changed pixels wins.

(e) Try to (very briefly, 2-3 sentences) formulate a theory about the underlying algorithm in the human visual system which makes it believe that B is brighter than A. Can you verify your theory on more examples? Do you believe it is more a "low-level" function, i.e. early in the processing pipeline, or a "high-level" one, i.e. requiring some insight into the scene?

Points: First, second and third place in the competitions for (c) and (d) will get 3,2 and 1 point(s), respectively. Part (e) will award another 4 points for a plausible explanation.