Computer Vision and Image Analysis
Assignment Sheet 2 - 04.11.2014

Next exercise group: 31.10.2014

- Deadline for exercise: **28.10.2014, 13:30**, 30% point loss per day late.
  Please submit solutions either on our Ilias web page (preferred method), or via e-Mail in a single ZIP-Archive to `ole.johannsen@uni-konstanz.de`.
- You may work in groups of up to three students, make sure all participants are clearly mentioned or assigned to the submission in Ilias.

Exercise 2.1 (image derivatives, about 10 points)
In this exercise a robust method to compute image derivatives should be introduced.

Given that additive noise is present in an image, $f = \hat{f} + \eta$, the discrete (forward) gradient computes as $\frac{\partial}{\partial x_i} f(i,j) \approx f(i+1,j) - f(i,j) = \hat{f}(i+1,j) - \hat{f}(i,j) + \eta_+ - \eta_-$. Unfortunately the noise does not cancel out but might add up to twice the amount of noise as in the original image. Thus, one idea to reduce the noise in the image gradient might be to combine the derivative of the image with a Gaussian kernel. This approach will be explored in this exercise.

1. One of the properties of the convolution is that $\frac{\partial}{\partial x_i}(f * g) = f * \frac{\partial g}{\partial x_i} = \frac{\partial f}{\partial x_i} * g$. Thus computing the derivative on an image convolved with a Gaussian kernel is equivalent to convolving the image with the derivative of the gaussian kernel which is equivalent to convolving the image derivative with a Gaussian kernel. Your first task is to calculate the derivative of the (continuous) gaussian function and discretise it to generate a kernel. Apply it to an image of your choice. Make sure the elements of the filters sum to zero. you should have calculated the derivative analytically... 2/4
2. Compare the convolution with the Gaussian derivative to the standard forward differences (e.g. calculated by *[f(:,2:end) f(:,end)]-f)*). Do so for different parameters of the Gaussian (σ as well as kernel size). Visualise so that gray represents intensity 0.5 (i.e. that the value 0 has the same colour in all figures). 0/4
3. Compute the magnitude of the gradient and visualise. 0/2

Exercise 2.2 (separable kernels, 2+4+6+2+2 points)
Let $f \in \mathbb{R}^{n \times m}$ be a filter kernel, written as a matrix, in this context it is not important where exactly the center is. The kernel f is called *separable* if it can be written as the convolution of a "vertical" kernel $h \in \mathbb{R}^{n \times 1}$ (i.e. a column vector) and a "horizontal" kernel $g \in \mathbb{R}^{1 \times m}$ (i.e. a row vector). In formulas:

Due to associativity of convolution, convolving an image with a separable kernel can be implemented as a row convolution followed by a column convolution, which is much more efficient than a single 2D convolution (see part (d)).

Your tasks are as follows.

(a) Let f and g be two separable filter kernels. Show that $f * g$ is separable.

(b) Let $f \in \mathbb{R}^{n \times m}$ be a filter kernel, written as a matrix. Show that f is separable if all singular values other than σ_1 are zero (σ_1 can be any value). What are the resulting row and column kernels?

(c) Write functions for row and column convolution, then extend your own convolution function from exercise 1(b) as follows:

- Check first whether the kernel is separable (to alleviate numerical errors during computation of the SVD, you might want to choose some threshold how large the second singular value is allowed to be).

- If it is separable, call the row and column convolution functions, otherwise call the old code.

(d) Perform convolution with your new function with separable kernels of different sizes and compare the performance to the old one.

(e) Test the performance of `conv2` with both separable and non-separable kernels, and make and educated guess about whether Matlab checks for separability before performing a convolution.

Exercise 2.3 (denoising competition, 10(+3) points)
This exercise will teach you a bit about benchmarks and how computer vision methods are usually tested. Producing competitive results on benchmarks is (maybe unfortunately) a vital skill if you want your papers published.

In the online material on Ilias, you will find three images:

- `rebecca_original.png`: A photo of a little girl
- `rebecca_noisy.png`: The same photo with noise added
- `lukas_noisy.png`: Photo of a boy with noise added

The two images of Rebecca will be your (very small) training set. The image of Lukas is noisy, but you do not know the original - only we do. However, we tell you that the type and statistics of the noise is exactly the same as for the Rebecca image.

Your tasks are now as follows.

(a) Combine denoising methods you have learned about in the lecture to denoise `rebecca_noisy.png`. Try to achieve a maximum PSNR when comparing your denoised image to the original. You do not have to implement everything yourself, but may look around online for additional MATLAB source code. A good idea is probably to look at the histogram of the noise, which you can obtain e.g. by subtracting the original from the noisy image - this might give you an idea of which of the methods might work well. After each step, you can also take a look at the histogram of the still remaining noise. It might be useful to optimally fine-tune parameters using a script. For this exercise, you get 10 points if you submit the source code to achieve a final PSNR of above 28 dB. 10/10

(b) Compute a denoised version of `lukas_noisy.png` and submit it. Since you do not know the original, you cannot compute PSNR for the result - but we can. The top three results will get 3,2 and 1 bonus point(s), respectively.