EXERCISES FOR IMAGE PROCESSING I

PROBLEM SHEET 4

Due date: 27.11.14 before 12:00h

Topics: Histograms, Filters, Convolution and Fourier-Transform

Submission: Please send your solutions via email to seppke@informatik.uni-hamburg.de.

1 THEORETICAL PROBLEMS 10 P.

a) Histograms and Noise:
   - Explain briefly, what a histogram is representing, and how it is created from the image. What kind of histogram do you expect for an image showing white parcels on a black conveyor belt?
   - Assume, that your camera is affected by sensor noise. Does that change the histogram, too? If yes, describe the changes and the arising problems.

a) Projections:
   How can projections be used to separate lines and characters in optical character recognition (OCR)?

b) Filters:
   Show the linear character of Gaussian averaging and the non-linear character of median filtering. In other words, show that $med[f_1(x) + f_2(x)] \neq med[f_1(x)] + med[f_2(x)]$ for arbitrary regions of pixels and two image brightness functions f_1 and f_2.

c) Convolution:
   The image B1 shows a bright square before a darker background (see figure on the right). Let B2 be the image, which results from convolving B1 with itself. Describe B2 in qualitative terms.

2 PRACTICAL PROBLEMS 10 P.

a) Greyvalue normalization:
Determine and implement a grayscale transformation that maps the darkest 5% of the image pixels to black, the brightest 10% of pixels to white (255) and linearly transforms the greyvalues of all remaining pixels between black and white.

b) Fourier-Transform:
You have a program, which allows the computation of 1D-FFT for 2^K values of a real discrete function. How can you make use of this function to compute the 2D-Fourier-Transform for an image with 512 x 512 pixels? How can you realize the back transformation?
Implement your solution with numpy.fft and compare the results with the available 2D-Fourier transform. To simplify the task, use greyvalue images in 2D-arrays.

Useful numpy.fft-commands
(from: http://docs.scipy.org/doc/numpy/reference/routines.fft.html):

- 2D-Fourier transform: fi = fft2(i)
- 1D-Fourier transform of row k: fz = fft(i[k:k+1, :])
- 1D-Fourier transform of column k: fs = fft(i[:, k:k+1])
- Inverse transform: ifft and ifft2
  (the output is a complex-valued numpy array with real and imaginary values)