Intelligent Systems Group
Universität Duisburg-Essen
Prof. Dr. Josef Pauli / Stefan Lörcks

WS 18/19
2018-10-19

Exercises Computer/Robot Vision - 01

PROGRAMMING (7 points)

The goal of this weeks programming exercises is to learn basics of image processing in MATLAB. This includes reading and writing of images, converting color images to gray-scale, simulating different kinds of noise and smoothing.

4. **Images: Read, show and write** (3 points)
   This exercises deals with the very basics of image processing in MATLAB: Loading, displaying and saving images.

   a) Create a script named

      `CRV_4.m`

      The first eight lines should look like:

```
%% CRV_04_ImagesReadShowWrite
% name: John Doe
% student number: 11235813

%% clean up
clear all;
close all;
clc;
```

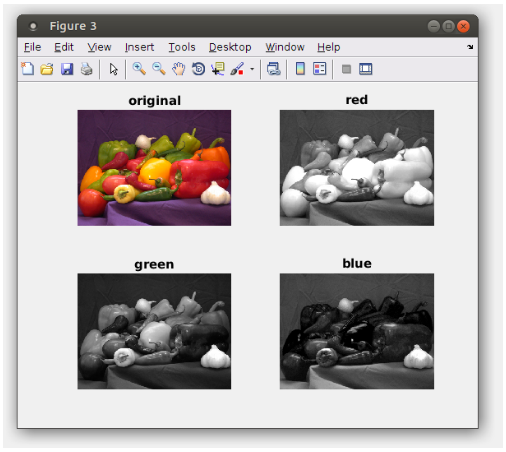
      Of course again, fill in your name and student number!

   b) In your script add a section named 'Gray-scale image'. Here you should load the image 'cameraman.tif', which is provided by MATLAB, into a variable in the workspace. Create a figure and show the image stored in the variable.
      Useful commands: `imread`, `figure`, `imshow`

   c) Add another section 'RGB image'. (RGB stands for red-green-blue. It is a common way to store color images.) In this section load the image 'peppers.png' into a workspace variable named 'rgbImage'. Create a second figure and show the image.

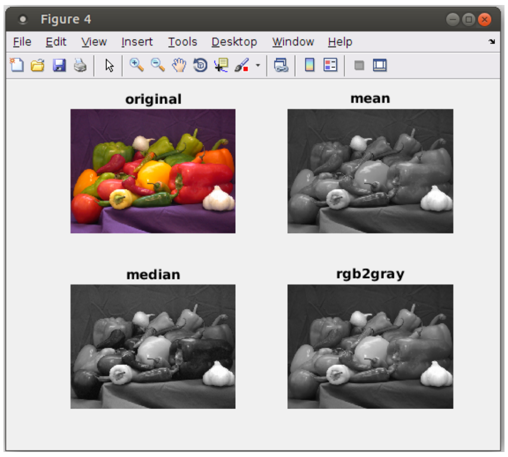
   d) RGB images are stored as a threedimensional arrays. The first and the second dimension are spatial while the third dimension is used for the color channels. This means you can extract the red channel just by indexing: `rgbImage(:,:,1)`.
      Extend the section 'RGB image' by creating a third figure, which shows the

color image and the three channels seperately. It should look like this:



Useful commands: `figure, subplot, title`

e) Within the framework of this course we will focus on algorithms and techniques working on gray-level images. If our input is RGB, we have to convert it into gray-scale first.
Create a section 'rgb2intensity'. Here, try out three different convertions: Averaging, median and the MATLAB built-in function `rgb2gray`. Show the results in a fourth figure:



Useful commands: `mean, median, rgb2gray, uint8`

f) Add a section 'saving'. Save the three gray-value versions of peppers (mean, median and rgb2gray) in the current directory. Use suitable filenames and the PNG format.
Useful commands: `imwrite`

5. **Smoothing, noise and denoising** (4 points)
In this exercise we will try out different ways to do gaussian smoothing, simulate white noise and impulse noise and use basic methods of denoising.

   a) Create a script named

   `CRV_5.m`

   The first eight lines should look like:

```
%% CRV_05_Smoothing, noise and denoising
% name: John Doe
% student number: 11235813

%% clean up
clear all;
close all;
clc;
```

   Of course again, fill in your name and student number!

   b) In your script add a section named 'Gray-scale image'. Here you should load the image 'cameraman.tif'.

   c) Create a section 'Gaussian operator'. Create a 3×3 matrix with the Gaussian operator via Pascal's triangle as introduced in the lecture. Normalize it, such that the 9 entries sum up to one. Compare the result to the output of `fspecial('gaussian',3,`σ`)` with different values for the parameter σ. Write down what you find out as a comment in the script.
   Useful commands: `fspecial`

   d) Apply both your own operator and the one created by `fspecial` on the image using the command `imfilter`, which implements the correlation. Show the original image and the two smoothed images within one figure using `subplot`. Add titles!
   Useful commands: `imfilter, figure, subplot, imshow, title`

   e) In image processing one often has to deal with noisy images. In MATLAB one can simulate noise with the command `imnoise`. For a gray-scale image stored in the variable `I` the command `imnoise(I,'gaussian')` adds Gaussian white noise (zero mean, variance 0.01) to the image.
   Create a section 'Gaussian noise'. Calculate a noisy version of the image. Show both the original and the noisy version in one figure. Add titles!
   Useful commands: `imnoise, figure, subplot, imshow, title`

   f) As pointed out in the lecture a way to deal with noise is gaussian smoothing. Despite the function `imfilter`, MATLAB also offers a function `imgaussfilt`. It is more sophisticated: The function decides, if the smoothing is done in the spatial domain or in the frequency domain (based on fourier transform). It

also sets the filter size automatically.
Extend the previously created section: Use `imgaussfilt` to calculate smoothed versions of the original image and the noisy image. Extend the figure, such that all four images are shown: The original one, the smoothed original, the noisy and the smoothed noisy image. Try out different values for the standard deviation in `imgaussfilt`.

g) Another type of noise that can occure in image processing is impulse noise (salt & pepper noise).
Create a section 'Impulse noise'. Spice up the image with some salt and pepper using `imnoise(I,'salt & pepper')`. Try to denoise the image by using `imgaussfilt`. Create a figure, which shows the original image, the smoothed original, the impulse noise image and the smoothed impulse noise image. Try out different values for the standard deviation in `imgaussfilt`.

h) Extend the previous section: Instead of gaussian smoothing, now try to remove the impulse noise with median filtering. `medfilt2` implements this. Extend the figure to show the result aswell. (The figure should now show five images.)
Useful commands: `medfilt2`