Intelligent Systems Group
Universität Duisburg-Essen
Prof. Dr. Josef Pauli / Stefan Lörcks
WS 18/19
2017-10-26

Exercises Computer/Robot Vision - 02

PROGRAMMING (7 points)

To get a better understanding of how correlation works, we will implement it. As an application of correlation we will perform edge detection. This will include denoising as well.

6. **Correlation** (2 points)
   In this exercise we will implement the correlation as introduced in the lecture.

   a) Create a function file named

   `myCorrelation.m`

   The header should look like:

```
function [ fimCorr ] = myCorrelation( fim, fop )
%MYCORRELATION calculation 2D correlation for CRV
%   fimCorr = myCorrelation(fim,fop) calculates the correlation of the
%   image function fim and the operator function fop as introduced in the
%   CRV lecture. Both fim and fop have to be matrices. fop has to be
%   smaller than fim and of odd size. fimCorr is of the same size as fim.
%   The outer pixels, where the correlation can not be calculated
%   sufficiently, are set to zero.
```

   This file can be found in the moodle course.

   b) Include a check whether or not both your inputs are two dimensional. If they are not (e.g. three dimensional) throw an error.
   Useful commands: `ndims`, `ismatrix`, `error`
   Example:
   `myCorrelation(rand(640,480), rand(3,3))` should be fine.
   `myCorrelation(rand(640,480,3), rand(3,3))` should result in an error.

   c) Determine the size of the operator pattern `fop`. Check, whether or not the operator pattern `fop` is of odd size. If its size is even in at least one dimension an error should be thrown.
   Useful commands: `size`, `mod`

Example:
`myCorrelation(rand(640,480), rand(5,3))` should be fine.
`myCorrelation(rand(640,480), rand(4,3))` should result in an error.

d) Check, whether or not the operator pattern `fop` is suffiently smaller than the image `fim`. This means `fop` has to fit in fim at least at one position, such that one value for `fimCorr` can be calculated. Otherwise an error should be thrown.
Example:
`myCorrelation(rand(640,480), rand(7,7))` should be fine.
`myCorrelation(rand(640,480), rand(501,501))` should result in an error.

e) Convert both inputs to floating point numbers with double precision to avoid datatype incompatibilities.
Useful commands: `double`

f) Calculate the half side lengths h_1 and h_2 of the operator pattern.

g) Implement the correlation. This includes creating the output variable `fimCorr`, calculating the inner values and setting the outer values to zero.
Example:

```
>> fim = magic(4); % A magic square
>> fop = kron([-1; 0; 1],ones(1,3)); % Vertical Prewitt
>> fimCorr = myCorrelation(fim,fop)

fimCorr =

     0     0     0     0
     0     1     7     0
     0     7     1     0
     0     0     0     0
```

h) Make sure everything is working as expected. The following examples should give the same results for your implementation of `myCorrelation`:

```
>> fimCorr = myCorrelation(magic(5),magic(3))

fimCorr =

     0     0     0     0     0
     0   405   570   585     0
     0   550   615   730     0
     0   595   760   575     0
     0     0     0     0     0

>> fimCorr = myCorrelation(magic(5), -1:1)

fimCorr =
```

```
         0   -16   -16    14     0
         0   -16     9     9     0
         0     9    14     9     0
         0     9     9   -16     0
         0    14   -16   -16     0

>> fimCorr = myCorrelation(magic(5),(-1:1)')

fimCorr =

         0     0     0     0     0
       -13   -18    12    12     7
       -13     7    12     7   -13
         7    12    12   -18   -13
         0     0     0     0     0
```

7. **Edge detection** (5 points)
In this exercise we will perform edge detection. First we will implement the edge detection methods, that have been introduced in the lecture. Afterwards we have a look at the edge detection methods MATLAB provides. In the end, we will try to do the best edge detection for eight test images. Pre-processing like denoising will be necessary.

   a) Create a script named

   `CRV_7.m`

   The first eight lines should look like:

```
1 %% CRV_07_EdgeDetection
  % name: John Doe
3 % student number: 11235813

5 %% clean up
  clear all;
7 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'. Convert it to floating point numbers and normalize it by dividing by 255.

   c) Create a section 'SimpleDerivation' in your script. Perform edge detection using the operator

$$f_{Gr_x}^{op} := \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$$

   and

$$f_{Gr_y}^{op} = \left(f_{Gr_x}^{op}\right)'$$

respectively.

i. Create the operator patterns.

ii. Calculate both parts of the gradient image $Gr_x^{im}(x,y)$ and $Gr_y^{im}(x,y)$ by correlating the image with the operator patterns. If you did the previous exercise you can use `myCorrelation`. Otherwise you can use `imfilter` or `filter2`.

iii. Calculate the magnitude of gradient

$$\mathrm{MoG}(x,y) := \sqrt{\left(Gr_x^{im}(x,y)\right)^2 + \left(Gr_y^{im}(x,y)\right)^2}$$

.

iv. Perform a thresholding with the threshold $\nu = 0.5$ to get the final binary edge map:

$$f_{edge}^{im}(x,y) := \begin{cases} 1 & \text{if } \mathrm{MoG}(x,y) > \nu \\ 0 & \text{if } \mathrm{MoG}(x,y) \leq \nu \end{cases}$$

v. Create a second binary edge map for the threshold $\nu = 1.5$.

vi. Create a figure. Show the original image, both parts of the gradient, the magnitude of gradient and both edge maps in the figure using subplot. This means your figure should contain six images. Add titles!

d) Add a section 'Prewitt' to your script. Repeat the steps from 7c) with the Prewitt operator and the thresholds $\nu = 0.5$ and $\nu = 1.5$.

e) Add a section 'Sobel' to your script. Repeat the steps from 7c) with the Sobel operator and the thresholds $\nu = 0.5$ and $\nu = 1.2$.

f) Add a section 'edge' to your script. Use the function `edge` provided by MATLAB to perform edge detection. `edge` provides several methods for edge detection. Try out Prewitt, Sobel and Canny. Use `doc edge` to find out how to use `edge`. Create a figure showing the original image and the binary edge images from `edge` for Prewitt, Sobel and Canny. Add titles.

g) Download the file edgetestimages.zip from the moodle course. It contains eight test images. Copy these pictures to your working directory. For each image create a section in your script and in there try to do the best edge detection you are able to do. You can use everything you learned so far. This includes smoothing, denoising, etc. and working with different parameters.