Intelligent Systems Group
Universität Duisburg-Essen
Prof. Dr. Josef Pauli / Stefan Lörcks
WS 18/19
2018-11-02

Exercises Computer/Robot Vision - 03

PROGRAMMING (7 points)

Import information
Write code that can be executed on any computer. Not just on the computer you used. This includes not making use of specific directories, e.g.

```
imread('C:\Users\Stefan\Uni\CRV\exercises\3\testimages\Test1.png')
```

Things like that will obviously not work during the grading. Unless stated otherwise assume that images you are reading are provided in your current working directory.

This exercise sheet deals with active contours. We will implement the shrinkage method, that has been introduced in the lecture. Before doing this main step, we will learn how to visualize a curve and how to create it from user input.

8. **Child's play - The house of St. Nicholas** (1 point)
   When dealing with active contours it is important to be able to visualize the results and intermediate steps. This means we have to able to draw the extracted contour on the image. In this exercise we will learn how to draw a curve on an image.

   a) Create a script named

   `CRV_8.m`

   The first eight lines should look like:

```
%% CRV_08_Child'sPlay
% 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' into the variable `I`.

   c) Add the following piece of code to the script and execute it:

```
%% Octagon
x = [224 196 128 60 32 60 128 196 224];
y = [128 196 224 196 128 60 32 60 128];
imshow(I);
hold on;
plot(x,y,'rx:','LineWidth',2);
hold off;
```

As you can see, this draws a octagon on the image. The coordinates of the vertices are given in the vectors `x` and `y`. Take into account that the last coordinates equal the first ones to create a closed curve.

d) In your script add a section named 'Haus vom Nikolaus'. Here create a second figure. Again, show the gray-scale image loaded in b). On this image draw the 'Haus vom Nikolaus' [1], a closed curve consisting of eight straight lines. This is a famous german child's play, which is also connected to graph theory. Choose different colors for the vertices and connecting lines.

9. **Curve input** (2 points)
Active contours require user interaction to define the initial curve. In this exercise we will learn how to define such a curve.

a) Create a script named

`CRV_9.m`

The first eight lines should look like:

```
%% CRV_09_CurveInput
% 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' into the variable `I`.

c) Open a figure and use the command `roipoly` to define a curve using the mouse cursor. The usage of `roipoly` is explained in the documentation. After the user finished drawing the curve with a doubleclick, extract the coordinates of the points/vertices of the curve and round them to integers. Also, close the figure.

d) Open a new figure. Show the image and visualize the curve you created using the integer coordinates. Use different colors for the points and the connecting straight lines.

10. **Active Contours - The shrinking method** (4 points)
In this exercise we will implement the shrinking method as introduced in the lecture.

a) Create a function file named

`MyActiveContour.m`

The header should look like:

```
function [ x, y ] = MyActiveContour( I, X0, Y0, N, SIGMA )
%MYACTIVECONTOUR performs simple contour extraction
%   [ x, y ] = MyActiveContour( I, x0, y0, N, sigma ) extracts the boundary
%   of a object. I is a intensity image. x0 and y0 are column vectors
%   containing the coordinates of the initial curves vertices. The last
%   vertex has to equal the first one. N defines the number of iteration. A
%   gaussian blur with variance sigma is applied on the image first.
```

This file can be found in the moodle course.

b) Smooth the intensity image `I` using a gaussian blur with standard deviation `sigma`.

c) Calculate the magnitude of gradient of the smoothed image using the Sobel operator.

d) Determine the energy component E_{ima} as introduced in the lecture. For some pixels the energy might be calculate to be ∞. Set these entries to 10000.

e) Determine the number of vertices in the initial curve. Keep in mind, that the last vertex is just a repetition of the first one.

f) Initialize the output variables with the coordinates of the initial curve.

g) Implement the actual shrinking method, that has been introduced in the lecture. In this approach each point p_i of the current curve can just be shifted one step into the direction of the successive point p_{i+1}. This is done for each point of the curve one after another. The current point p_i is just shifted to its neighbour point $p_i^{'}$, if the energy does not increase.
In the original approach this is done for all points of the curve over and over again until the energy does not change anymore. Instead of this, here we do this for a fixed number of cycles given by `N`. In pseudocode this looks like:

Algorithm 1 Shrinking method

```
for 1 ... N do
    for every point in current curve do
        Determine coordinates of current point p_i
        Determine coordinates of neighboring point p_i'
        if energy of p_i' is not higher than energy of p_i then
            Move p_i to p_i'
        end if
    end for
end for
```

If you updated the coordinates of the first point of the curve also make sure to update the last coordinates!

Hint:

An easy way to determine the coordinates of the neighbouring point $p_i^{'}$ of the current point p_i that is located into the direction of the successive curve point p_{i+1} is the following:

i. Determine the vector u from the current point p_i to its successor p_{i+1}.

ii. Determine the signs of the entry of that vector.

iii. The coordinates of $p_i^{'}$ are given by adding these signs to the coordinates of the current point p_i.

h) Try out, if your function is working fine using the follow test scenario, which is available on moodle ('MyActiveContourTesting.m'):

```
>> I = ones(200);
I(40:60,140:160) = 0;
x0 = [35 45 55 65 65 55 45 35 35]';
y0 = [145 135 135 145 155 165 165 155 145]';
[ x, y ] = MyActiveContour( I, x0, y0, 50, 0.1 )

x =

     40
     60
     60
     60
     60
     40
     40
     40
     40
```

```
y =

   140
   140
   140
   160
   160
   160
   160
   140
   140
```

i) Create a script named

`CRV_10.m`

The first eight lines should look like:

```
%% CRV_10_MyActiveContour
% name: John Doe
% student number: 11235813

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

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

j) Create a section 'Test1' in your script. Read the image Test1.png (from your current working directory), which is provided in the moodle course in TestImages.zip. Perform contour extraction on the image.

   i. Let the user create a initial curve.

   ii. Use your function `MyActiveContour` to calculate the boundary. Try out different parameters `N` and `sigma`.

   iii. Create a figure. Use `subplot` to show the image with the initial curve and the image with the final curve. Add titles!

k) Add a section 'Test2' to your script. Repeat the steps from 10j) with the image 'Test2.png'.

l) Add a section 'Test3' to your script. Repeat the steps from 10j) with the image 'Test3.png'.

m) Add a section 'coins' to your script. Repeat the steps from 10j) with the image 'coins.png', which comes with MATLAB.

References

[1] Wikipedia. *Haus vom Nikolaus — Wikipedia, Die freie Enzyklopädie*. 2018. URL: `https://de.wikipedia.org/w/index.php?title=Haus_vom_Nikolaus&oldid=177260455`.