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

WS 18/19
2018-11-09

# Exercises Computer/Robot Vision - 04

## 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\4\houghtestimages\edge01.png')
```

Things like that will obviously not work during the grading. Therefore, load all images from the current working directory. Do not make use of any specific subfolders.

This exercise sheet deals with the Hough transform. We will implement a simple version of the Hough transform and visualize the results.

11. **Hough transform for straight lines** (7 point)
In this exercise we will implement a simple version of the hough transform to detect straight lines using the polar form. It will be able to detect straight lines. Flexibility regarding the resolution of the feature space or computional effiency are not the focus of this exercise.

    a) Create a function file named

    `MyHough.m`

    The header should look like:

```
function [ akku, h, alpha ] = MyHough( edgeImage )
%MYHOUGH detects straight lines vie Hough transform
%   [ akku, h, alpha ] = MyHough( edgeImage ) calculates the Hough
%   transform of the binary edgeImage. The hough image is contained in
%   akku. h is a vector containing the distance values. With D being the
%   diagonal length of the edgeImage, the values in h range from -D to D.
%   alpha contains the angle values in degree. These range from -90 to 89.
```

    This file can be found in the moodle course.

b) Round up the diagonal length of the edge image to determine `D`. Create the vectors `h` and `alpha`. They should range like announced in the header. For both a stepsize of one is fine.

c) Create the hough akkumulator.

d) Implement the Hough transform: For every edge point in edge image:

- Determine the coordinates of the edge point.
- Determine all possible combinations of `h` and `alpha` that lead to an edge point at that position/coordinates.
- Increase the akkumulator at the corresponding positions.

Useful commands: `cos`, `find`, `length`, `round`, `sin`
Forbidden commands: `hough`, `houghlines`, `houghpeaks`, etc.

e) Try out, if your function is working fine. As a first step use the follow test scenario, which is available on moodle ('MyHoughTesting.m'):

```
%% Create edge image
edgeImage = zeros(100);
edgeImage(30,:) = round(rand(100,1));
edgeImage(60,:) = round(rand(100,1));

%% Hough transform
[accu, h, alpha] = MyHough(edgeImage);

%% Visualization
figure();
subplot(131);
imshow(edgeImage);
title('edge image');
subplot(132);
imshow(accu,[]); % ,[] enables automatic scaling
axis square;
title('hough image');
subplot(133);
imshow(log(1+accu),[]);
axis square;
title('hough image (logarithmic scale)');
```

The output should look like:


Once again the results depends on the choice of the coordinate system. Here, the MATLAB way of indexing has been used.

f) Create a script named

`CRV_11.m`

The first eight lines should look like:

```
%% CRV_11_MyHough
% name: John Doe
% student number: 11235813

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

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

g) Create a section 'edge01'.

i. Load the edge image edge01.png, which is available on moodle in (HoughTestImages.zip). Convert it to double and make it a binary image containing zeros and ones.

ii. Perform a Hough transform using `MyHough`.

iii. Create a figure showing the edge image and the hough image.

iv. Determine the highest peak in the hough image and the corresponding values of $h$ and $\alpha$.
Useful commands: `max`

v. Draw the corresponding straight line in the edge image.
Useful commands: `plot`, `fimplicit` + Anonymous functions

vi. Extend your code to be able to determine the $N$ highest peaks in the hough image and drawing all corresponding lines in the figure.

vii. Set $N$ to the lowest possible values, which still leads to a detection of all four sides of the rectangle.

h) Create a section 'edge02'. Repeat the steps of 11g) with the edge image edge02.png. Here set $N$ to the lowest value that still leads to a visualization of all four lines.