Week 1 Assignments

Introduction
During this week you will be working with different types of image processing operators to get a feel how things work. You will be using an online resource for this. In the second assignment this week you have to segment characters and digits from standard yellow Dutch license plates.

Exercise 1 – Image processing basics
In this exercise you will be experimenting with different image processing tools. You are expected to use ImageJ, an open source image processing tool written in Java.

a) Go to http://rsbweb.nih.gov/ij/ and download and install ImageJ.
b) Take some pictures and process them by using
   - thresholding,
   - histogram equalization,
   - minimum filter
   - maximum filter
   - median filter
   - average filter
   - Find edges using the Laplace and Sobel operators

   Note 1: if the transformation uses the intensity levels in the neighborhood, then the original image needs to be processed with a 3*3 and a 9*9 neighborhood filter.

   Note 2: in addition to the lecture slides you can also read the material on http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm.

Deliverables: The written report needs to contain the input images and the transformed images. For each transformed image a short discussion of the results needs to be given.

Exercise 2 - Dutch license plate segmentation

In this exercise you have to make a Java program that can segment license plates from a picture. Segmentation is the process of finding the area of interest in the picture and generating, in this case, six images which contain a digit or letter from the license plate. You will be given a partial solution which must be completed by you. Next week you are asked to create a full license plate recognizer.

a) Unpack "licenseplates.rar" in a suitable directory. Note that the directory "licenseplates/inbox" contains several license plate pictures. Take at least ten pictures of Dutch (yellow!) license plates and place them in "licenseplates/inbox". Remark that the pictures need to be in jpg format.

b) Open the NetBeans project "DLP Student" (Dutch Licence Plates). Study the code. The variable "workDir" in the Runner class needs to contain the name of the directory in which the "licenseplates.rar" was unpacked, see a).

c) Explain the name ChannelFilterThreshold.

d) Implement the rgb value in the method filter() in ChannelFilterThreshold. Hint: How is the color yellow constructed from red, green and blue? Test it and compare it to plate_filter22.bmp found in "exampleoutput.rar".

e) Implement the class Segmenter. The purpose of this class is to create six ordered images representing the digits and characters on the license plate from left to right on basis of the filter image. In exampleprocess.zip you can find plate_1seg22.bmp to plate_6seg22.bmp as an expected output from this step.

   One, slightly, naive way to solve this problem is to make two assumptions :

   - The yellow color of the license plate has turned into the **largest black blob** after the filter method from ChannelFilterThreshold is applied, for example have a look at plate_filter22.bmp. A blob is a collection of points with the same color (black or white in this case) and each member of the blob is related to all the other members of the blob by using a connected relation (in directional terms N, E, S, W, NE, SE, SW and NW, also called 8-connectivity) or is transitively connected. The latter means that if A and B are connected and B and C are connected, then A and C are also connected. This step can create another intermediate image, look at plate_clean22.bmp in "exampleoutput.rar". This assumption can be implemented in the method largestBlob(). An example algorithm is described in: http://www.labbookpages.co.uk/software/imgProc/blobDetection.html#code
   - The characters and digits that have to be recognized are the **largest six white blobs within the largest black blob**. After the six segments have been found, they still have to be placed in order, i.e. from left to right.

That is easily done by looking at the x coordinates. This assumption can be implemented in the method createSegments().

The source code you are given uses these assumptions.

Please note that using the two assumptions is just a suggestion. Note that these assumptions are not always met, so the algorithm might create some bizarre segmentation! If you come up with a much nicer, more robust and faster algorithm you are more than welcome to implement it ☺!

f) In order to make the segmentation slightly more robust the class Segmenter contains an extra method enhance() that generates another intermediate image before the six blobs are found. Study its implementation, explain the threshold and look carefully at the difference between plate_cleanup22.bmp and plate_enhance22.bmp. Why would one use such an enhancement? Hint: why would somebody use this *before* finding the six blobs?

Deliverables: In addition to the source code only the answers to question c) and f) needs to be handed in on paper. I will ask you to process some pictures and explain the various intermediate results. Furthermore you need to be able to explain the source code.