Lab 1: Matlab Introduction/Histograms

In this lab we will cover some basics of Image Processing in Matlab. The task to be performed is loading an image into Matlab and converting the image data to another color representation (e.g. from RGB to rgb). Secondly, you will be introduced to histograms for object representation. The Mean-Shift tracking algorithm uses a histogram for object representation, so this is one of the parts of the final implementation.

For image processing, you can find brief introduction in the blackboard. The command to read an image from disk is *imread*. The formula for the rgb color space can be found in the syllabus. Matlab performs best while avoiding *for* loops. So operations should be applied on images as a whole. Use logical arrays to deal with division-by-zero (don't turn off the warnings).

Regarding the remark not to turn off warning messages, but to solve the problems, we will put some emphasis on creating nice, robust and beautiful code. As opposed to ugly hacks. So use long and descriptive variable names, don't reuse names in confusing manners and pass all arguments of a function explicitly (i.e. no global variables).

Example images can be found in the directory **data** in the blackboard, but you can also use your own images. The goal of this exercise is to experiment with the different color models; see what problems arise when you try to convert the images to these different color models and try to find the advantages and disadvantages for these models.

Several color representations are discussed in the reader. For those of you who don't have this reader right now, the survey can be found here.

After reading an image from disk with *imread*, the data is of type *uint8.* This means that are values of the image are stored as integer-type, and not as floating point values. Before doing anything with this image, it is necessary to convert the image to type *double*. The easiest way to do this, is by using the command *im2double*, which scales the values between 0 and 1. Note that omitting this step will not cause an error in the subsequent calculations, but the values will simply be rounded off to the nearest integer value. In the case of converting to normalized-RGB, this means that all values will become either 0 or 1!

The next task is to experiment with histogram creation and histogram back projection. You can use the image of Nemo in the **data** folder (or any other image, for that matter) as test images. For instance, as a first step, try to create a representation of Nemo, and see if histogram back projection can find the location of Nemo in this image.

Histograms are frequently used for image and object representation, as you might know. A histogram contains information about the colors present in an object. Histograms are of course not limited to color, other aspects of objects can also be used in the histogram. For this lab we will assume color histograms, but you are free to experiment with other features for your final application.

A histogram can be seen as a (quantized) representation of the colors in an image, without any of the spatial information. But histograms are also a way to represent statistical distributions, representing the distribution of colors in a certain object. Seen in this light, the values in the histogram bins represent the likelihood that a color occurs in the represented object.

To restrict the histogram to an object instead of a whole image, you could perform several tricks. You could crop the image and create an image containing only the object. Or you could use the masking that you also used above. This mask could be filled parametrically (using some sort of function are inequality, e.g. `x < 10 || x > 213`). Another option is to open the image in an editing program, e.g. Paint, and paint the object white and the rest of the image black, and create a mask from this image.

Sometimes the result of this procedure can be improved by weighting the histogram with a histogram of background objects (i.e. everything but the target). This histogram is simply the ratio of the target histogram divided by the non-target histogram. The result of this procedure is that color values, which are also prominent in the background, will have a lower value. Some experimenting is needed to see if this theoretical improvement also leads to real-life improvements.

So, to summarize, for this week's assignment, you should:

- Read an image from disk, and write functions to convert this image from one color space into another.
- Implement color space conversions for RGB to normalized-RGB, RGB to Opponent Color Space and RGB to HSV.
- For normalized-RGB, how do you handle the corner cases (e.g. when R, G and B are all 0? Or close to 0?).
- Implement a function to create a 2D- or 3D-histogram of your image, depending on the color space you are using. For instance, RGB requires a 3D-histogram, normalized-RGB, i.e. rgb, only needs a 2D-histogram (why?). Try visualizing your histogram to verify that it is correct.
- In the opponent color space, negative values can occur. Why?