Problem Set 01

1. (Basic Acquaintance with Programmed Imaging) Implement a program (e.g., in Java, C++, or Matlab) that does the following:

1.1 Load a color (RGB) image I in a lossless data format, such as `bmp`, `png`, or `tiff`, and display it on a screen.

1.2 Display the histograms of all three color channels of I.

1.3 Move the mouse cursor within your image. For the current pixel location p in the image, compute and display:

a) The *outer border* (see grey box) of the 11 x 11 square window W_p around pixel p in your image I (i. e., p is the reference point of this window).
b) (Above this window or in a separate command window) the location p (i. e., its coordinates) of the mouse cursor and the RGB values of your image I at p.
c) (Below this window or in a separate command window) the intensity value $[R(p) + G(p) + B(p)]/3$ at p.
d) The mean μ_{W_p} and standard deviation σ_{W_p}.

1.4 Discuss examples of image windows W_p (within your selected input images) where you see "homogeneous distributions of image values", and windows showing "inhomogeneous areas". Try to define your definition of "homogeneous" or "inhomogeneous" in terms of histograms, means, or variances.

> The *outer border* of an 11 x 11 square window is a 13 x 13 square curve (which could be drawn, e. g., in white) having the recent cursor position at its center. You are expected that you dynamically update this outer border of the 11 x 11 window when moving the cursor.
>
> Alternatively, you could show the 11 x 11 window in a second frame on a screen. It is also encouraged to look for solutions that are equivalent in performance (same information to the user, similar run time, and so on).

2. (Data Measures o Image Sequences) Define three different data measures $D_i(t)$, i = 1, 2, 3, for analyzing image sequences. Your program should do the following:

a) Read as input an image sequence of at least 50 frames;
b) Calculate your data measures $D_i(t)$, i = 1, 2, 3, for those frames;
c) Normalize the obtained functions such that all have the same mean and the same variance;
d) Compare the normalized functions by using the L_1-metric.

3. (Different Impacts of Amplitudes and Phase in Frequency Space on Resulting Filtered Images) It is assumed that you have access to FFT programs for the 2D DFT and inverse 2D DFT. The task is to study the problem of evaluating information contained in amplitude and phase of the Fourier transforms:

a) Transform images of identical size into the frequency domain. Map the resulting complex numbers into amplitudes and phases. Use the amplitudes of one image and the phases of the other image, and transform the resulting array of complex numbers back into the spatial domain. Who is "winning", i.e. can you see the image contributing the amplitude or the image contributing the phase?
b) Select scalar images showing some type of homogeneous textures; transform these into the frequency domain and modify either the amplitude or the phase of the Fourier transform in a uniform way (for all frequencies), before transforming back into the spatial domain. Which modification causes a more significant change in the image?
c) Do the same operations and tests for a set of images showing faces of human beings.

Discuss your findings. How do uniform changes (or different degrees), either in amplitude or in phase, alter the information in the given image?

4. (Approximating the HSI Space by Planar Cuts) Assume that G_{max} = 255. We are cutting the RGB cube by a plane Π_u that is orthogonal to the grey-level diagonal and passing through the grey level (u,u,u) for $0 \le u \le 255$. Each cut (i.e., the intersection of Π_u with the RGB cube) is represented by one N x N image I_u, where the value $\mathbf{u} = (R,G,B)$ at pixel location (x,y) is either defined by the nearest integer-valued RGB triple in the RGB cube (or the mean if there is more than one nearest RGB triple), if this distance is less than $\sqrt{2}$, or equals a default value (say, black) otherwise. Do the following:

a) Implement a program which allows one to show the RGB images I_u, for $u = 0$, $u = 1, \ldots,$ $u = 255$ (e.g., by specifying the value of u in a dialogue or by having a continuously running animation);
b) Also show the (scalar) saturation values instead of RGB values. Figure 1 shows the results for $u = 131$;
c) You may either select a fixed value N > 30 (size of the image), or you may also allow a user to choose N within a given range.

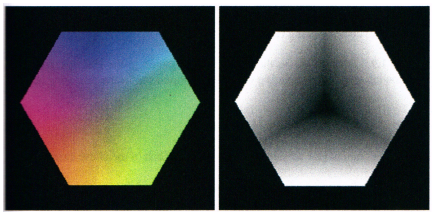

Figure 1. Cuts through the RGB cube at showing the RGB image and saturation values for the same cutting plane.