# Laboration 3

Bildreproduktion och Bildkvalitet
(TNM097)

---

Bildkvalitet

# Goal:

This lab is to gain a deeper understanding of a few important quality metrics by using them to evaluate a number of images.

After this lab, you will gain a better understanding of:

- Color gamut
- Mathematical quality metrics, such as MSE and SNR
- The human visual system (HSV) models and why they are needed
- S-CIELab quality metric
- SSIM quality metric

# Introduction:

In this lab, you will test a number of quality metrics on different grayscale and color images. You will also study if or under what circumstances they correlate with your own visual judgment of the quality. All metrics in this lab have been discussed in lecture 3 and can be found in the corresponding lecture notes and literature at Lisam.

All images and files you will need you are available in Lab3.zip ans scielab.zip in Lisam, under course documents/Labs/Lab3.

**Don't forget** to scale all images to the interval [0, 1] after you read them in Matlab, for example by *im2double*.

# 1. Color Gamut:

Here you are supposed to plot the color gamut for two different devices in the same plot. You can use the function *plot_chrom(XYZ, col)*, which takes the CIEXYZ data for the device primaries and a string *col* to specify the color of the graph. Notice that in this function it is assumed that XYZ is a 3 x T matrix, where T is the number of the primaries of the device. Therefore, each column contains the CIEXYZ values for a primary.

**Device 1:** A Dell computer screen. The CIEXYZ data of the RGB-primaries for this screen is saved in *Dell.mat*.

**Device 2:** An Inkjet printer (Canon IPF 6450) using seven primary inks, C, M, Y, K, R, G and B. The CIEXYZ data of the primaries for this printer is saved in *Inkjet.mat*. In order to easily plot this the data in the chromaticity diagram, the CIEXYZ for black is not included in this data.

Discuss the differences between these two devices and try to figure out which device is better in reproducing a specific color, for example red.
……
……
……
……
……

# 2. Mathematical metrics

## *2.1 Grayscale image (MSE / SNR)*

Start by reading a grayscale image (for example peppers_gray.tif) and test one of the metrics *MSE* or *SNR* for two different "distortions". You can either use Matlab's pre-defined function *snr* or the function *mysnr* provided to you, **just note that in both of them the first argument**

**is the original signal/image and the second argument is the noise** (**noise:** the difference between the original and the reproduction).

### 2.1.1 Interpolation

Use the function *imresize* to down-sample and then up-sample the image by 0.25 and 4, respectively. Use three different interpolations, "*nearest*", "*bilinear*" and "*bicubic*".

**Hint**: *imresize(imresize(p,0.25,'nearest'),4,'nearest')*; does the intended operation using nearest neighbor interpolation.

Look at these three images and apply MSE or SNR to them to find how similar they are to the original image. Does the result of these metrics correlate with your judgment of the quality?

……
……
……
……
……

### 2.1.2 Halftoning

Halftone the grayscale image by thresholding it with 0.5 and also error diffusion (use Matlab function *dither*).
**Hint**: a >= 0.5, thresholds image *a* with 0.5.
**Hint**: The resulting halftones will be of class *logical*, make them double in order to be able to use them in your calculations.

Look at these two halftones and apply *MSE* or *SNR* to them to find how similar they are to the original image. Does the result of these metrics correlate with your judgment of the quality? Is there any halftone that is more similar to the original than the image thresholded with 0.5 according to these two metrics? Why?

……
……
……
……
……

## *2.2 Color image ($\Delta E_{ab}$)*

Start by reading a color image (for example *peppers_color.tif*) and test the $\Delta E_{ab}$ metric for two different halftones. Notice that the RGB color images have to be converted to CIEXYZ and then to CIELab. As you know the conversion from RGB to XYZ is device dependent and therefore in this lab you can use one of the three conversions you tried for the input device in Lab1, i.e. assignment 3.2, 3.4 or 3.5 in Lab1. **However, in this lab it is ok to use the predefined function *rgb2lab* in Matlab:**

*imLab=rgb2lab(imrgb);*

where *imrgb* is the input image (rgb) and *imLab* is the output image consisting of L, a and b channels, where the first, second and the third channel in imLab contain L, a and b values, respectively.
Halftone the color image by thresholding it with 0.5 and also error diffusion (use Matlab function *dither*). **Note** that in this case the r, g and b channels are first halftoned separately and then merged together to make the color halftones.
Look at these two halftones and apply $\Delta E_{ab}$ metric to find how similar they are to the original image. Does the result of these metrics correlate with your judgment of the quality? What halftone is the most similar to the original according to these two metrics?

……
……
……

# 3. Mathematical metrics involving HVS

You are given a function *snr_filter*, which is the SNR function modified by a model for HVS. Make sure that you understand how this function operates by reading the comments and also having a look at the other functions that are called inside this function.
Apply this metric to the two halftones in assignment 2.1.2. Does the result correlate better with your judgment of quality?

……
……
……

Extend now the $\Delta E_{ab}$ metric used in assignment 2.2 by involving the HVS model (function *MFTsp*) you are given. In order to make it simple, you just need to apply the HVS model to R, G and B channels separately before converting them to CIEXYZ and then CIELab. **Note** that after filtering the R, G and B channels you might end up with negative R, G or B values. Make sure that the negative values in these matrices are replaced by zeros before converting them to XYZ. This can be done for the R channel by for example *R=(R>0).*R* and similarly for the other two channels. **It is ok to use the predefined function *rgb2lab* in Matlab.**

Now apply your modified $\Delta E_{ab}$ metric to the two halftones in assignment 2.2. Does the result correlate better with your visual judgment of quality?
……
……
……

**Note** that this metric is not completely correct because you applied the same filter to the channels in the RGB space. The correct metric would be to apply different filters to different channels in the opponent color space (S-CIELab metric), used in the next section.

# 4. S-CIELab

Among the files put at Lisam you have a file called *scielab.zip*, in which you have all necessary codes to run the function called *scielab*. For more details, please read about the method and the codes at: http://white.stanford.edu/~brian/scielab/.

## *4.1 S-CIELab as a full-reference metric*

The function *scielab* has a number of optional arguments. To use it as a full-reference metric in this lab you need to input five arguments as follows,
The first argument is *sampPerDeg* (sample per degree). Please read the link above (or have a look at the lecture notes) to understand how this value is related to the device resolution and the viewing distance. The second and the third arguments are the original image and the reproduction in the CIEXYZ space, respectively. Therefore, you first need to convert the RGB images to CIEXYZ, for example by the predefined Matlab function *rgb2xyz*. The fourth argument is a vector containing the CIEXYZ values of the white point of the intended light source, in the case of CIED65 it is [95.05, 100, 108.9]. This is to compute the CIELab values. The last argument (fifth) is the image format, which in this lab is 'xyz'. The output of this function is an image the same size as the input images containing the S-CIElab difference between the two images. In order to have a quality value you can use the average value of the output image. The less this value, the smaller the difference between the original and the reproduction.
Now read the color image *peppers_color.tif*. Down-sample and up-sample (using *imresize*) the image by 0.25 and 4, respectively, using all three interpolations. Look at the results and compare them using S-CIELab metric. Notice that in order to have a fair judgment you need to know the screen resolution of your screen and also specify a viewing distance. Does the metric correlate with your judgment of the quality? Study the image of S-CIELab difference and discuss the results.
……

…….
…….
…….
…….

## *4.2 S-CIELab as a no-reference metric*

To use *scielab* as a no-reference metric in this lab you just need to input two arguments as follows,
The first argument is *sampPerDeg* (sample per degree). The second argument is the reproduction, in the CIEXYZ space. Therefore, you first need to convert the RGB image to CIEXYZ. The output of this function is an image the same size as the input image containing three channels, CIEL, a and b. The sum of the standard deviations of these three channels could be used as a metric for graininess. The smaller the standard deviation, the less the graininess. The function *std2(A)* returns the standard deviation of the elements in the matrix *A*.
In this section, we want to test this metric to evaluate the graininess of a number of color halftones. Load the file *colorhalftones.mat* in Matlab. You will have five color halftone patches, called c1 through c5.
c1 and c2 are halftone patches containing cyan and magenta dots. While in c1, the cyan and magenta channels have been halftoned independently, in c2 the channels have been halftoned dependently. Look at these two halftones and test the S-CIELab metric to judge the graininess of these two patches. Does the result correlate with your judgment?
…….
…….
…….

c3, c4 and c5 are halftone patches containing cyan, magenta and yellow dots. In c3, all three channels have been halftoned independently. In c4 all channels have been halftoned dependently. In c5 only cyan and magenta channels have been halftoned dependently and the yellow channel independent of the two others. Look at these three halftones and test the S-CIELab metric to judge the graininess of these three patches. Does the result correlate with your judgment?
…….
…….
…….

# 5. SSIM

In this section, we want to test the image quality metric SSIM (structural similarity), for which there is a predefined function *ssim* in Matlab. As the input this function takes the original image and the reproduction (or the distorted image). As the output this function returns a structural similarity value and a map/image including the local SSIM value for each pixel.
Read the grayscale image, *peppers_gray.tif*. In order to test this metric, we simulate a number of distortions and use *ssim* to calculate the similarity between the original and the distorted image.

**a-** ***distortion 1*****:** add +0.1 to odd rows (i.e. rows: 1, 3, 5, etc.) and -0.1 to the other rows (i.e. rows: 2, 4, 6, etc.) in the original image. ***distortion 2*****:** add 0.1 to the upper half of the original image and subtract 0.1 from the lower half of it. Both distortions will exactly give the same SNR when compared to the original image (check it), why? Now use the function *ssim* to check how similar these two distorted images are to the original. Look also at the SSIM map. Comments?
…….
…….
…….
…….
…….

**b-** ***distortion 1*****:** add noise to the original image, for example with dist1=orig+0.2*(rand(size(orig))-0.5), where *orig* is the original image. ***distortion 2*****:** Create a

Gaussian filter by f=fspecial('gauss',21,10) and filter the original image by this filter. Use SNR and SSIM metric to see how similar these two images are to the original. Look also at the SSIM map. Comments?

……
……
……