COMP.SGN.240 Advanced Signal Processing Laboratory

Laboratory Exercise - Digital Image Formation and Processing

Contacts: andrea.corsini@tuni.fi, lucio.azzari@tuni.fi, alessandro.foi@tuni.fi

This exercise is part of the Advanced Signal Processing Laboratory course. Prerequisites are basics of Image Processing and MATLAB. The exercise must be done in groups of one or preferably two students.

1 General Instructions

In this exercise, raw images taken with a digital camera are analyzed and processed in Matlab. This exercise will familiarize the group with basics of digital image acquisition, formation, and processing, including analysis of common degradations such as noise. The exercise also helps the group to understand the main steps in an image processing pipeline of a digital camera.

A brief survey of an image acquisition pipeline for digital cameras is:
Color Image Processing Pipeline [A general survey of digital still camera processing], by R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew [1].

Summary

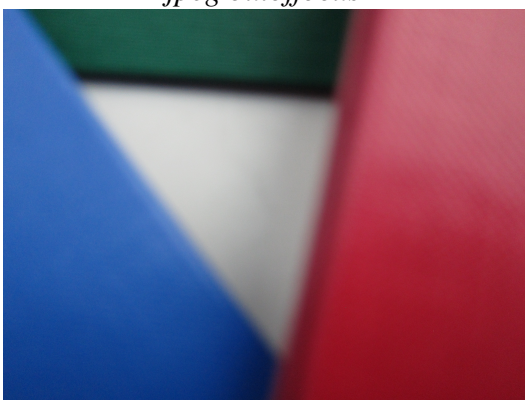
- Write MATLAB functions for the tasks listed in Section 2.
- Test your functions carefully, and implement a main function that goes through all the tasks and does not require input parameters. Ask for help by email if needed.
- Write a technical report explaining in detail the functions implemented. The report must also include the answers to the questions of Section 2.
- Return your MATLAB codes and the report in the Moodle.

To learn the syntax of Matlab functions it is recommended to consult the Matlab Documentation. Just execute the command `doc` in the Matlab command window and the Documentation Center will start. To look for the syntax of a specific function you can also execute directly the command `doc function_name`.
SUGGESTION: you can also check the syntax of a function directly on the Command Window by using its *help*. To visualize a function's help execute the command `help function_name`.

2 Exercise Instructions

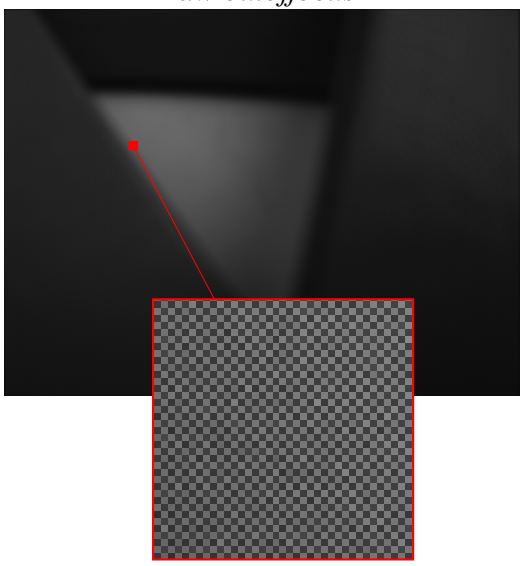
The provided material will include two *raw* images: an out-of-focus (blurry) image (*outoffocus.tiff*) with low contrast that will be used to estimate the noise statistics, and a natural image (*natural.tiff*) that will be processed and enhanced.
For each block-wise operation on the images it is RECOMMENDED, but it does not influence the grade, the use of the command `blockproc`.

jpeg outoffocus

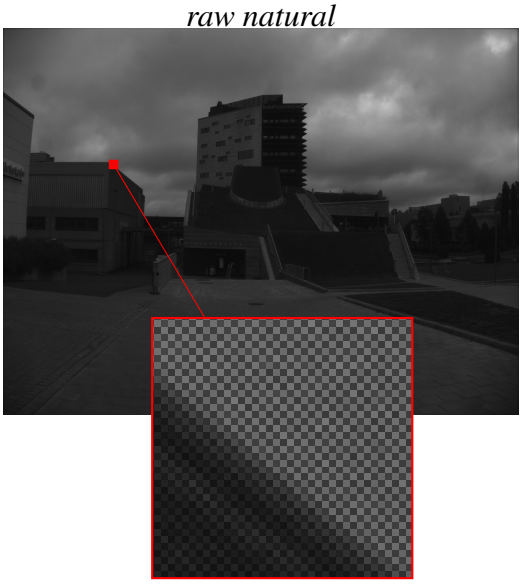


jpeg natural

raw outoffocus



raw natural



Tasks:

1. Load the images in Matlab and convert them to double.
2. Visualize the images and illustrate the Bayer mosaic arrays.
3. Separate the images into subchannels (generally referred to as I_c, $c = 1, \ldots, 4$), and visualize them separately. Assume that the pattern is RGGB (see the Wikipedia page to check different color scheme examples).

4. Analyze each subchannel of the **out-of-focus** image with a sliding window operator that outputs the local sample means and local sample variances for each position of the window. Create *mean-variance* scatterplots (*i.e* scatterplots where the coordinates of each point in the plot are the sample mean and sample variance over a window) to visualize the signal-dependent variance within each subchannel.

5. Fit straight lines (*i.e.* an affine variance relation) to the mean-variance scatterplots of the **out-of-focus** image, and plot them against the corresponding scatterplots. Let the fitted lines have the form

$$\text{<noise-variance>} = a_c \cdot \text{<signal-mean>} + b_c, \tag{1}$$

where the subscript $\cdot_c$ indicates the respective color channel c.

The parameters a_c and b_c describe the variance of the noise affecting I_c in function of the noise-free image (here called signal-mean). Note that since the estimated line is not flat, *i.e.* $a_c \neq 0$, the variance of the noise depends on the signal that the noise is contaminating. This type of noise is commonly referred to as *signal-dependent*, and it is common to most of the digital imaging acquisition systems.

6. Apply the transformation

$$2\sqrt{\frac{I_c}{a_c} + \frac{3}{8} + \frac{b_c}{a_c^2}} \tag{2}$$

to each subchannel of the **natural** and **out-of-focus** images, using the corresponding parameters a_c and b_c estimated from the **out-of-focus** image.

7. Compute the mean-variance scatterplots of the transformed **out-of-focus** image and compare it to the scatterplots from the original **out-of-focus** image. The new scatterplots should now be flat (or almost), and equal to the constant 1. This happens because (2) is a variance stabilizing transformation that makes the resulting noise variance *signal-independent*, constant, and unitary, *i.e.* $a_c \approx 0$ and $b \approx 1$. The transformation (2) is a generalization of the Anscombe transform [2].

8. Implement in Matlab a *sliding-DCT filter* (*e.g.*, the one from DCT Image Denoising: a Simple and Effective Image Denoising Algorithm, by G. Yu and G. Sapiro [3]), and filter both **natural** image and its transformed version. Select the threshold parameter λ that yields the most visually convincing results. Apply the filter to each subchannel separately.

9. Apply the inverse transformation

$$a_c \left(\frac{1}{4} D^2 + \frac{1}{4}\sqrt{\frac{3}{2}} D^{-1} - \frac{11}{8} D^{-2} + \frac{5}{8}\sqrt{\frac{3}{2}} D^{-3} - \frac{1}{8} - \frac{b_c}{a_c^2} \right),$$

where D denotes the filtered transformed image.

10. Compare the filtered **natural** image and the one obtained by the inverse transformation of the filtered transformed **natural** image. Why was it useful to apply the root transformation and inverse? Observe that the inverse transformation is not precisely the inverse function of the forward root transform; speculate on why it is done so.

11. Perform simple demosaicking: interpolate the images using `interp2` and `meshgrid` in such a way to take into account the relative shift between the channels present in the Bayer array.

12. Compose a color image and display it. Note that the displayed image is not similar to jpeg shown above; this happens because white balancing has been applied to the image above. Apply white balancing to the image.
    Hint: a simple and effective white balancing method is to 1) project the image into a color space composed by [1 luminance + 2 chrominances], such as HSV, YUV, *etc.*; 2) find the luminance pixel with maximum intensity; 3) divide each color channel of the RGB image by the corresponding R, G, or B values that the maximum pixel has. In this way the maximum of the image will be displayed as white, *i.e.* $(1, 1, 1)$.

13. Perform contrast and saturation correction in your favourite color space (*e.g.*, HSV, YCbCr) taking advantage of, *e.g.*, the histogram equalization function of Matlab, gamma-correction, or other.

References

[1] Ramanath, R., Snyder, W.E., Yoo, Y. and Drew, M.S. "Color image processing pipeline." IEEE Signal processing magazine 22.1 (2005): 34-43. https://doi.org/10.1109/MSP.2005.1407713

[2] Anscombe, F. J. "The statistical analysis of insect counts based on the negative binomial distribution." Biometrics 5.2 (1949): 165-173.

[3] Yu, G. and Sapiro, G. "DCT Image Denoising: a Simple and Effective Image Denoising Algorithm." Image Processing On Line 1 (2011): 292-296. https://doi.org/10.5201/ipol.2011.ys-dct