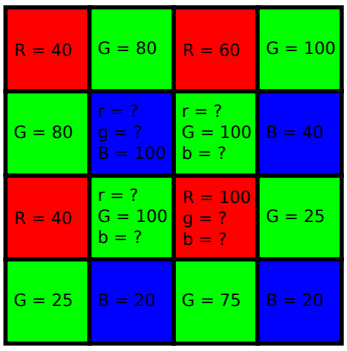
Assignment 3

Welcome to the third assignment of the lecture *2D Vision and Deep Learning*. **Please read all instructions carefully!** The goal of this assignment is to familiarize you with the image formation process and spatial image filtering.

Submission is due on Monday, November 23rd , 2020 at 2pm. Please note that late assignments will receive zero (0) marks, so you are strongly encouraged to start the assignment early. If you have any questions please contact Tim Heußler (`theussle@students.uni-mainz.de`).

Exercise 1 (3 points)**.** Which is the distance of the pixels with coordinates $(17, 42)$ and $(289, 68)$ using the *4 neighborhood*, the *8 neighborhood* or *euclidean distance*?

Exercise 2 (2 points)**.** Determine the unknown color parts of the following sensor endowed with a Bayer filter using bilinear interpolation.



Exercise 3 (2 points)**.** Show that cross correlation is *not associative* (use e.g. $\mathbf{I}^E = [1\ 2\ 1]$ and $\mathbf{I}^{F_1} = \mathbf{I}^{F_2} = [-1\ 0\ 1]$).

Exercise 4 (2 points)**.** Calculate the convolution between the filter matrix $\mathbf{I}^F$ and the image $\mathbf{I}^E$.

$$\mathbf{I}^F = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad \text{and} \quad \mathbf{I}^E = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array} \text{ (periodically continued)}$$

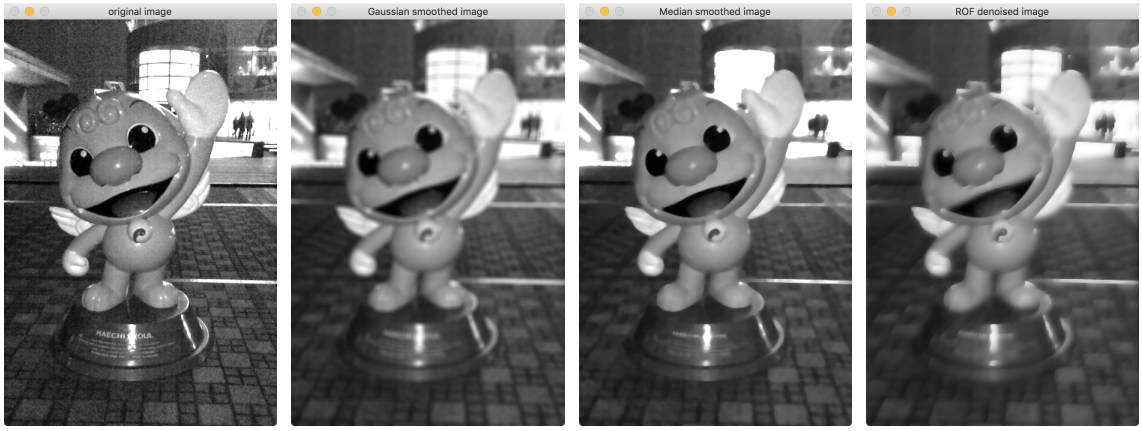
Exercise 5 (7 points)**.** Write a simple OpenCV python script that reads an image and applies

1. (3 points) the *low-pass filters* `GaussianBlur(...)`, `bilateralFilter(...)` and `medianBlur(...)`. Thereby, it should be possible to adjust the different parameters using *sliders*.

2. (2 points) the *high-pass filters* `Sobel-` bzw. `Scharr-filter`. Thereby, your program should display (a) an image that shows the image gradient in x direction, (b) an image that shows the image gradient in y direction, and (c) an image that shows the length of the image gradient.

3. (2 Punkte) a *Canny edge filter* (`cv2.Canny(...)`). *Compare the result* with the image showing the length of the image gradient determined with the help of a *Sobel-* or *Scharr-filter*.

Exercise 6 (3 points)**.** Write a simple OpenCV python script that reads an image and applies a denoising filter to the image that minimizes the *total variation (TV)* of the image, i.e. the sum of the gradient norm $J(I) = \int |\nabla I| d\mathbf{x}$. A python implementation of this idea (*Rudin-Osher-Fatemi de-noising model (ROF)*), described in *A. Chambolle et al., Total Variation Minimization and a Class of Binary MRF Models, EMMCVPR 2005*, can be found in *J. E. Solem, Programming Computer Vision with Python, O'Reilly, 2012*. Used OpenCVs method `imshow(...)` to display the result. The following pictures compare different denoising techniques.



Exercise 7 (10 points)**.** Write a python script that can rectify images. Your program should implement a user interface that allows to select four points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ in an image with the mouse. After the four points have been selected, it should be possible to move the points to new positions $\mathbf{p}'_1, \mathbf{p}'_2, \mathbf{p}'_3, \mathbf{p}'_4$ with the mouse. The mapping between these two pointsets determine a *homography* H, i.e. a 2D projective transformation that maps points in one plane to another. This homography can now be used to eliminate perspective distortions, such as the distorsion of the lettering *Schwarz* in the image below. Use the OpenCV functions `findHomography(...)` and `warpPerspective(...)` to realize such an image rectification.