STB600 Lab 2: Intensity transformation and spatial filtering

Atieh Sahraeidolatkhaneh: atieh.sahraeidolatkhaneh@hv.se

Yongcui Mi: yongcui.mi@hv.se

1. Objectives

The objective of this lab is to practice with image intensity transformation and spatial filtering.

2. Tasks

Apply these different intensity transformation and spatial filtering methods to the given image. Discuss and comment on the obtained results, such as what are happened to the image by there operations, when these methods can be used, advantages and disadvantages of these methods.

2.1. Intensity transformation

For the task of intensity transformation, you can work with image provided *img.tif*.

2.1.1. Image negatives

Use built-in function 'cv2.bitwise()' in OpenCV to invert every bit.

```
import cv2

# Read image

# Invert the image

# Show the image
```

2.1.2. Log Transformations

Log transformation of an image means replacing all pixel values in the image with its logarithmic values. The formula for log transformation is as follows:

S = c * log (1 + r)

r = input pixel value

c = scaling constant, can be given by 255/(log (1 + m)), where m is the maximum pixel value in the image.

S = output pixel value

```
import cv2
import numpy as np

# Read image

# Apply log transformation method

# Specify the data type

# Display both images
```

2.1.3. Power-Law (Gamma) Transformations

Power-law (gamma) transformations can be mathematically expressed as $s = cr^{\gamma}$. Apply gamma correction with at least 2 values for each of the two caess (a), 0¡gamma¡1; (b)gamma¿1. (Hint: gammaImg = np.array(255*(youImg / 255) ** gamma)

```
import cv2

# Read image

# Apply gamma correction

# Save & show the edited images
```

```
#You need to do
```

2.2. Spatial filtering

We will use 3 types of spatial filtering techniques in OpenCV. You can work on the provided images *gaussianNoiseImg.tif* and *peppersaltImg.tif*.

2.2.1. Averaging

Averaging is done by convolving the image with a normalized box filter. It simply takes the average of all the pixels under kernel area and replaces the central element with this average. This is done by the function cv2.blur() or cv2.boxFilter(). You should specify the width and height of kernel.

```
import cv2
import numpy as np

# Read image

# Apply the filter with 5x5 or 3x3 kernel size

# Show both images
```

2.2.2. Gaussian Filter

In this approach, instead of a box filter consisting of equal filter coefficients, a Gaussian kernel is used. It is done with the function, cv2.GaussianBlur(). You should specify the width and height of the kernel which should be positive and odd.

```
import cv2
import numpy as np

# Read image

# Apply Gaussian filter with kernel size 5x5

# Show both images
```

2.2.3. Median Filtering

The function cv2.medianBlur(inputImg, kernelSize) computes the median of all the pixels under the kernel window and the central pixel is replaced with this median value. This is highly effective in removing salt-and-pepper noise.

```
import cv2
import numpy as np

# Read image

# Apply Median filter with kernel size 5

# Show both images
```

```
#You need to do
```

3. Additional resource

References