Lab 3: Image Processing

CSE 185 Introduction to Computer Vision, Fall 2025

Description: In this lab, we will implement basic image processing operations including histogram equalization, image denoising via 2D convolution, and edge detection. We will get familiar with Numpy, OpenCV, and MatPlotlib library through this lab.
Instructions: Please use either Google Colab (as learned in the previous chapter) or a locally configured Python environment to complete this assignment. After finishing, **submit one PDF file using the submission template (PDF or Word) provided in the same zip file**, which should include:

- **Report**
- **Code text of Python source file(s) or screenshots of the Colab** `.ipynb` **notebook**

The final submitted PDF should be named **CSE185_Lab3_Submission_Name_StudentID.pdf**.

1 Histogram equalization

Histogram equalization is an image enhancement method by "stretching" the histogram of a low-contrast image to be a nearly uniform histogram, resulting in a high-contrast image. Follow the following steps for implementing histogram equalization.

1. Compute and visualize histogram and cumulative distance function (CDF) of an input gray-scale image
2. Apply histogram equalization using obtained CDF on the input image
3. Compute and visualize histogram of output image

Input: Use bay.png, which is included in the same zip file as this PDF.

```
%%% Useful code snippet:
% read an image as gray-scale image
import cv2
img = cv2.imread(path_to_image, cv2.IMREAD_GRAYSCALE)
% compute image histogram
hist,bins = np.histogram(img.flatten(),256,[0,256])
% compute cummulative density function
cdf = hist.cumsum()
cdf_normalized = cdf / cdf.max()
plt.plot(cdf_normalized)
```

2 Image denoising

Image denosing is to reduce noise in images. We will add common types of noise to an image and use 2D convolution for denoising. Follow the following steps.

1. Read the input image and convert to a grayscale image
2. Add two types of noise, including Gaussian noise and Salt/Pepper noise (2 separate images with distinct blur. Implement your own functions to add noise to an image.)
3. Implement mean and median filtering in 5x5 windows
4. Check if mean or median filtering is able to completely remove Gaussian noise or Salt/Pepper noise. Compare original image and denoised image.

Input: Use lena.png, which is included in the same zip file as this PDF.
(Lena.png is a popular image in the filed of image processing. `https://en.wikipedia.org/wiki/Lenna`)

```
%%% Useful code snippet:
% convert a color image to a grayscale image
lena_gray = cv2.cvtColor(lena, cv2.COLOR_BGR2GRAY)
% 2D convolution
image_out = cv2.filter2D(src=image_in, ddepth=-1, kernel=filter)
% median filtering
image_out = cv2.medianBlur(image_in, window_size)
```

3 Image gradient

We will compute image gradient and its magnitude to find image edges.

1. Read the input image and convert to a grayscale image
2. Compute image gradient in x and y direction respectively
3. Compute magnitude of image gradient for each pixel
4. Thresholding on magnitude to determine image edges, try various thresholds (no less than 3 thresholds).

Input: Use lena.png, which is included in the same zip file as this PDF.