2110431 Introduction to Digital Imaging

2147329 Digital Image Processing and Vision Systems

Homework #1

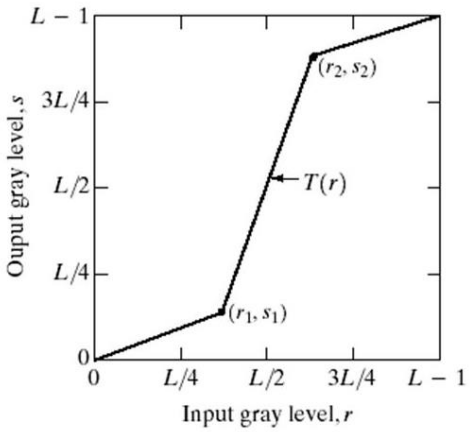
Deadline: September 5th, 2023 @23:59

Submissions: (1) PDF version of this file

(2) .ipynb file; template in this link:

https://colab.research.google.com/drive/1jub1fLi4a16kmD5JB2mhJdCPJKGOSkxw?usp=sharing

1. Write a program in python into **`homework1_1()`** function in **`homework1.ipynb`** file to implement contrast stretching follow the transformation in the graph below.



Test your program using kitty.jpg and your own image and display your results in the blank below.

Results of the processed images:

Original	After filtering
Kitty.jpg 	
Your image: 	

2. Design your own filter on an RGB image. Write your code in **`homework1_2()`** function in **`homework1.ipynb`** file Provide motivation behind the designed filter. Display it in terms of an RGB image.

Idea / Motivation:

I interest about detection people face from the image using image processing, so I try design a simply filter that can show only a face and decrease lightness of background in the image.

Your filter design (at least two equations and/or conditions):

For each pixel, it checks a condition based on the red channel's intensity compared to the intensities of the green and blue channels. If the red channel intensity is less than 1.2 times the intensity of either the green or blue channel, the pixel is considered not predominantly red. In this case, the code converts the pixel to grayscale by calculating dot product with [0.2989, 0.5870, 0.1140], resulting in a desaturated appearance and divided by 1.5 for make background darker (by using divided make the image look more flatten but I want to see the different between background and subject). But If that pixel is not in the condition (not predominantly red), apply a gamma factor of 0.8 to control the darkness level, rescaling the result, and ensuring it remains in the range [0, 255].

Examples of filtered image:

Original	After filtering
	
	

3. Two images, `f(x,y)` and `g(x,y)`, have histograms h_f and h_g. Write a program to display the histograms h_f and h_g. Then implement the operations below and display the new histogram of the output of each operation. Determine the new histograms in terms of h_f and h_g and explain how to obtain the histogram in each case (Optional).

(a) `f(x,y) + g(x,y)`

(b) `f(x,y) - g(x,y)`

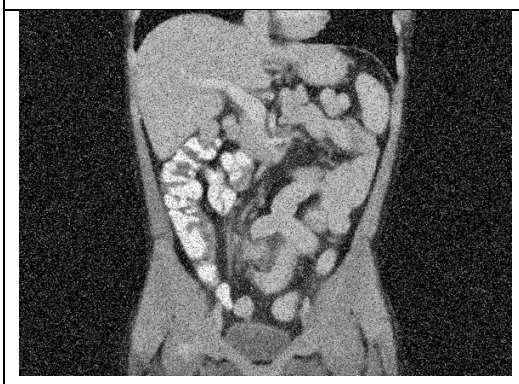
(c) `f(x,y) x g(x,y)`

(d) `f(x,y) ÷ g(x,y)`

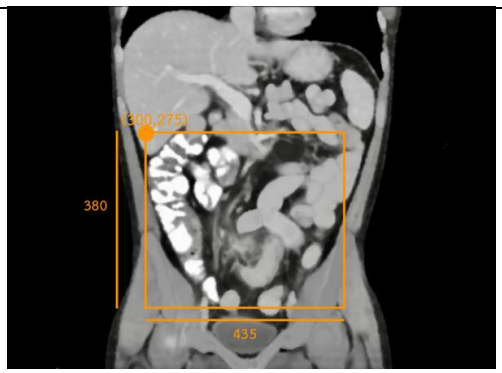
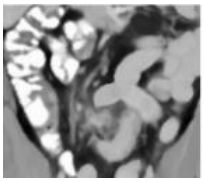
Hint: design one of the images very simple

4. Assume you work in the field of image processing. Your boss has assigned you a task to detect malignant tumors (assuming in this case, in bright intensity) from CT-SCAN images. The pain point is that the doctors saved images from the CT-SCAN, but the output images are incomplete and have Salt and Pepper noise. Please help the doctor remove the noise.

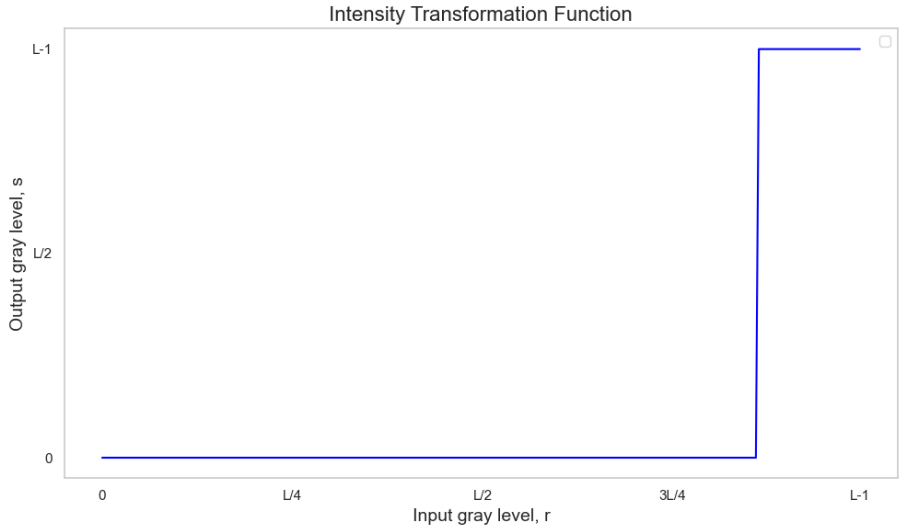
4.1 Apply a filter to remove the noise and select the appropriate size of the kernel. Provide your filtered image into the blank box below. Hint cv2.medianBlur

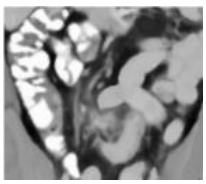
Original	After Filtering
	

4.2 Apply Region of Interest (ROI) with width=380 and height=435 start at x=300, y=275 as shown in the orange rectangle below and provide the ROI image in the blank box below. Hint cv2.rectangle

Hint	Mark ROI Image
	

4.3 Apply the transformation function shown in the graph below on the ROI image. This transformation function is used for segmenting malignant tumors (assuming, in this case the higher intensity) and show in a white mask. Provide the final segmented tumors in the blank box below.

Transformation Function

L-1 is the highest intensity level.
You can estimate the stepping point from the graph.

Original (Mark ROI)	Segmentation with Transformation
	

Download ipynb template in this link and submit your own ipynb in MCV (Not accept a link):

https://colab.research.google.com/drive/1jub1fLi4a16kmD5JB2mhJdCPJKGOSkxw?usp=sharing

Your code (in **homework1.ipynb**):

```
# import libraries here
import cv2
import numpy as np



def homework1_1(image_grayscale):
    # input -> image_grayscale - type -> np.ndarray, size of - (height, width)
    # output -> image_grayscale - type -> np.ndarray, size of - (height, width)

    # TO DO - Implement transformation based on the contrast stretching graph


def homework1_1(image_grayscale):
    # input -> image_grayscale - type -> np.ndarray, size of - (height, width)
    # output -> image_grayscale - type -> np.ndarray, size of - (height, width)

    # TO DO - Implement transformation based on the contrast stretching graph

    filtered_image = np.zeros_like(image_grayscale)

    def pixelVal(pix, r1, s1, r2, s2):
        if (0 <= pix and pix <= r1):
            return (s1 / r1)*pix
        elif (r1 < pix and pix <= r2):
            return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
        else:
            return ((255 - s2)/(255 - r2)) * (pix - r2) + s2

    (x,y) = image_grayscale.shape

    r1 = 56
    r2 = 169
    s1 = 0
```

```
    s2 = 255

    for x1 in range(x):
        for y1 in range(y):
            filtered_image[x1,y1] = pixelVal(image_grayscale[x1,y1], r1,
s1, r2, s2)

    return filtered_image


def homework1_2(rgbimage):
    # input -> rgbimage - type -> np.ndarray, size of - (height,
width, 3)
    # output -> filtered_image - type -> np.ndarray, size of -
(height, width, 3)

    # TO DO - Design your own filter
    filtered_image = rgbimage.copy()



    (w,h,d) = filtered_image.shape

    # print(filtered_image.shape)



    for x in range(w):

        for y in range(h):

            if(filtered_image[x,y,0] < filtered_image[x,y,1]*1.2 or
filtered_image[x,y,0] < filtered_image[x,y,2]*1.2):

                filtered_image[x,y] =
(np.dot(filtered_image[x,y],[0.2989, 0.5870, 0.1140]))//1.5

            else:

                filtered_image[x,y] = np.power(filtered_image[x,y] /
255.0, 0.8) * 255.0

                filtered_image[x,y] =
filtered_image[x,y].astype(np.uint8)



    return filtered_image
```

Your code (in **homework1.ipynb**):

```
# import libraries here
import cv2
import numpy as np
import matplotlib.pyplot as plt


def Home_work1_4 (rgbImage):
    # Convert the image to grayscale
    grayImg = cv2.cvtColor(rgbImage, cv2.COLOR_BGR2GRAY)

    # 4.1 Use the median filter to smooth the image
    smoothed_img = cv2.medianBlur(grayImg, 5)

    # 4.2 Make ROI with
    # Create an interesting area
    mark = np.ones((smoothed_img.shape[0], smoothed_img.shape[1]),
dtype=np.uint8) * 255
    # Set the coordinates of the rectangle
    start_point = (300, 275)
    end_point = (735, 655)

    # Draw a black square at the center
    cv2.rectangle(mark ,start_point, end_point, (0), -1)
    # Mark the area in the image
    mark_ROI = cv2.bitwise_or(smoothed_img, mark)

    # 4.3 Use Gray Level slicing
    Gray_Level_img = mark_ROI.copy()
    Gray_Level_img[(mark_ROI >= 223)]  = 255
    Gray_Level_img[(mark_ROI < 223)] = 0

    # Picture show Row1
    plt.figure(figsize=(20, 20))
    plt.subplot(1, 2, 1)
    plt.imshow(rgbImage[:, :, ::-1])
    plt.title('Original')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(smoothed_img, cmap='gray')
    plt.title('Image after smoothing')
    plt.axis('off')

    # Picture show Row2
    plt.figure(figsize=(10, 8))
    plt.subplot(2, 2, 1)
```

```
    plt.imshow(mark_ROI, cmap='gray')
    plt.title('Mark ROI Image')
    plt.axis('off')

    plt.subplot(2, 2, 2)
    plt.imshow(Gray_Level_img, cmap='gray')
    plt.title('Segmentation with transformation img')
    plt.axis('off')

    plt.show()
```