Middle East Technical University Department of Computer Engineering

CENG 466

Fundamentals of Image Processing

Fall '2021-2022

Take Home Exam 1

Due date: November 29 2021, Monday , 17:00

1 Objectives

The purpose of this assignment is to familiarize you with the fundamental spatial domain image enhancement techniques. For each question you are required to develop your own algorithm based on the techniques you learned in the lectures.

2 Specifications

You are given three questions, which you should solve with your own algorithms. In addition to the solutions, you are required to prepare a report that explains your methodology and includes the analysis of the results and your comments on them. The report should be **maximum 10 pages** long and should be prepared in IEEE Conferance Proceedings Template (LaTeXis recommended) provided in the following link.

`https://www.ieee.org/conferences_events/conferences/publishing/templates.html`

- Grading will be based on the quality of the outputs, script contents and the report
- The report should clearly explain the methodology and rationale behind the algorithm design. It should also explain the difficulties encountered in the design, implementation and experimentation stages, and your solutions on them. Last but not least, the report should contain your comments on the results. Even if the results does not match your expectations you should discuss the encountered situation.
- In the following sections you will be asked to implement several functions. Implement them in a single file called ***the1.py***. This file can contain any additional functions. Submit this file with your report. Do not submit input or output images.

3 Histogram Processing (35 Points)

In this part you will implement histogram matching algorithm on gray-scale images yourself. You will match the histogram of a given image to mixture of Gaussians. In order to complete this task follow the given steps:

Step 1: Define a function whose input will be the path of the original image, a list of the mean values, and a list of the standard deviations of the Gaussians.

```
def part1(input_img_path, output_path, m, s):
    '''
    Reads the image I from input image path
    Generates histogram with given mean and std-dev lists
    Equalizes the histogram of I with the generated histogram/
    Returns processed image

    input_img_path (str): Path to the image
    ex: './THE1_images/Part1/A1.png'
    output_path (str): Path where you should save your output images.
    ex: './THE1_outputs/Part1/'
    m (list): List of the means of gaussians
    ex: [0,1]
    s (list): Standard deviation of gaussians
    ex: [0.1, 2.5]
    '''

    return processed_img
```

Step 2: Read the input image from the given path. (You can use any python libraries such as cv2 or PIL). Note that these images will be provided in gray-scale.
Create the output directory (output_path) if it does not exist. (You can use os for this purpose).

Step 3: Extract the histogram of the given image. You should extract the histogram yourself. Do not use available functions. Save this histogram as an image to output_path with the name ***'original_histogram.png'***.

Step 4: Generate the histogram using Mixture of Gaussians with the given means and deviations. Recall the formula for Mixture of two Gaussians;

$$P(X) \sim \mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2).$$

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2},$$

where μ and σ are mean and standard deviation of the Gaussian distribution, respectively.

Hint: Assuming there are N pixel values in the given image, you can draw N random samples from this distribution. You can generate the histogram from those values.

You can use available python functions for this part. However, you should carefully explain them in your report.

Save this generated histogram as an image to output_path with the name ***'gaussian_histogram.png'***. An example is given in Figure 1.

Step 5: Match the histogram of the given image to the generated gauusian distribution. You should implement histogram matching yourself.
Save new image to output_path with the name ***'matched_image.png'***.
Save the histogram of the output image to output_path with the name ***'matched_image_histogram.png'***.

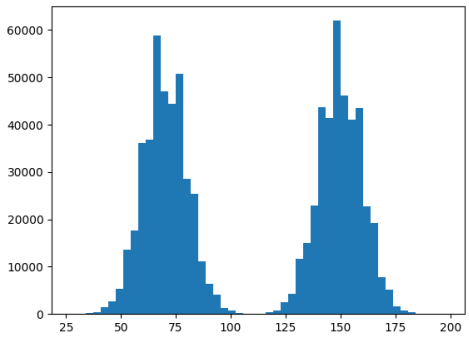


Figure 1: An example of a generated histogram

Step 5: **TESTING:** Use provided images 1.png and 2. png (see Figure 2a and 2b) to test your function with different parameters. Use the examples given below;

```
part1('./1.png', './Outputs/1/ex1', [45,200], [45,45])
part1('./1.png', './Outputs/1/ex2', [50,200], [10,10])
part1('./2.png', './Outputs/2/ex1', [75,200], [40,40])
part1('./2.png', './Outputs/2/ex2', [70,150], [20,20])
```

You should try at least 1 different mean-deviation combination per image. Make sure to use meaningful parameter sets.

Step 6: **REPORT:** Report and discuss your findings.

4 Spatial Domain Image Filtering (65 Points)

In this part you will be implementing spatial domain filtering. You have three different tasks to solve.

4.1 Convolution (15 Points)

In this part you will implement convolution function yourself. In order to complete this task follow the steps below:

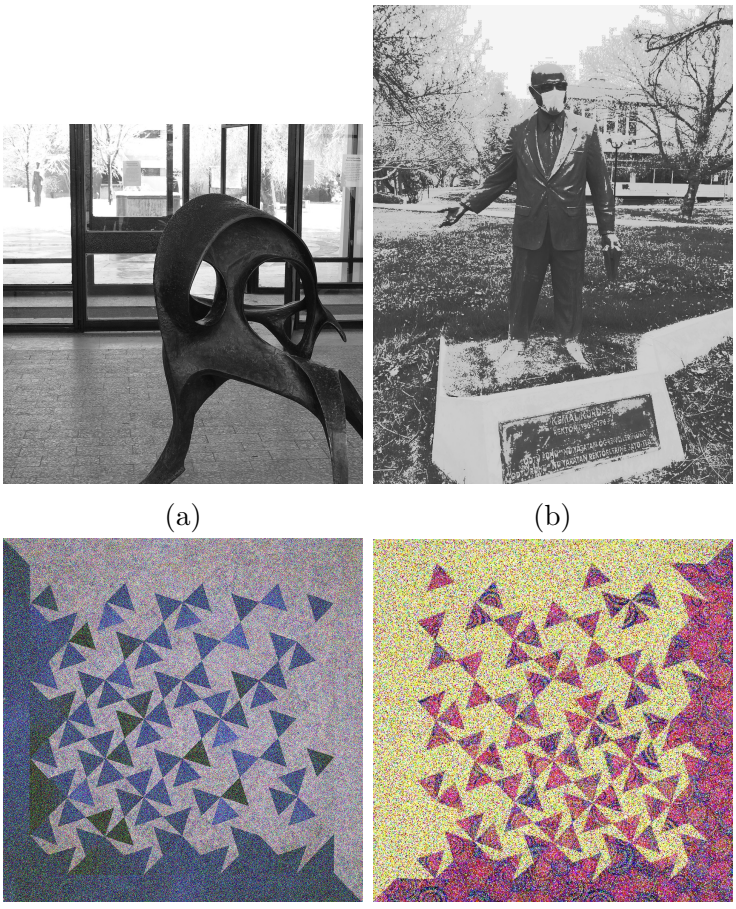
Step 1: Define a function whose input will be the path of the original image, and a filter.

```
def the1_convolution(input_img_path, filter):
    '''
    Reads the image I from input image path
    Applies convolution with the given filter
    Returns processed image

    input_img_path (str): Path to the image
    ex: './THE1_images/Part1/A1.png'
    filter (list): Given spatial domain filter
    ex: [[1,1,1],[1,1,1],[1,1,1]]
    '''
    return processed_img
```

Step 2: Read the input image from the given path. (You can use any python libraries such as cv2 or PIL).

Step 3: Apply convolution on the given image and return the processed image.



(a) (b)

(c) (d)

Figure 2: Given images for THE1. a and b are provided in gray-scale, whereas c and d are RGB images

4.2 Edge Detection (20 Points)

In this part you will implement an edge detection algorithm of your choosing. In order to complete this task follow the given steps:

Step 1: Define a function whose input will be the path of the original image, and output path as before.

```
def part2(input_img_path, output_path):
    '''
    Reads the image I from input image path
    Applies edge detection
    Returns the edge map
    '''

    return processed img
```

Step 2: Read the input image from the given path. (You can use any python libraries such as cv2 or PIL). Note that the images will be provided in gray-scale in this question. Create the output directory (output_path) if it does not exist. (You can use os for this purpose).

Step 3: Apply any edge detection algorithm of your choosing. It is highly recommended to use your own convolution function where needed.

Step 4: Extract the edge map of the given image. Save the output as an image to output_path with the name ***'edges.png'***.

Step 5: **TESTING:** Use provided images 1.png and 2. png (see Figure 2a and 2b) to test your function.

Step 6: **REPORT:** Report and discuss your findings.

4.3 Noise Reduction (30 Points)

This part is different than the previous ones, since in this question you will work on specific images. You will not write generic functions.
You are given two noisy images named as 3.png and 4.png (see Figures 2c and 2d). Your task is to enhance these images. For the sake of easy evaluation please follow the steps below;

Step 1: Define two functions whose inputs will be the path of the images(3.png and 4.png), and the output path

```
def enhance_3(path_to_3, output_path):
    '''
    Read image 3.png from input image path
    It will be given such as './THE1_images/3.png')
    Apply image enhancement
    Return the enhanced image
    '''
def enhance_4(path_to_4, output_path):
    '''
    Read image 4.png from input image path
    It will be given such as './THE1_images/4.png')
    Apply image enhancement
    Return the enhanced image
    '''
```

Step 2: Read the input image from the given path. (You can use any python libraries such as cv2 or PIL). Create the output directory (output_path) if it does not exist. (You can use os for this purpose).

Step 3: Analyze the type of the noise in the images and design a series of filters to denoise them. Note that you should examine each channel of the images. You an use any combination of filters.
You are free to use multiple algorithms for comparison.

Step 4: Save the enhanced image output_path with the name ***'enhanced.png'***.
(If you use multiple algorithms for comparison, you can save the outputs as ***'enhanced1.png'***, ***'enhanced2.png'*** etc.)

Step 5: **REPORT:** Report all the filters you have used and their purposes. Discuss the differences among filters according to your findings.

5 Regulations

1. **Group:** You are required to do your assignment in a group of two students. If there is an unclear part in your code, we may ask any of the group member to describe that code segment. Also group members may get **different** grades. We reserve the right to evaluate some or all of the groups to determine the contribution of each group member to the assignment.

2. **Programming Language:** You must code your program in Python. Your submission will be tested on department lab machines. You are expected make sure your code runs successfully on department lab machines.

3. **Late Submission:** Late Submission is **not** allowed!

4. **Newsgroup:** You must follow the odtuclass for discussions and possible updates on a daily basis.

6 Submission

Submission will be done via Odtuclass. Submit 'the1.py' (which includes all your functions), and your report **Do not send the input and output images.** Only one member should submit the homework. Hence, do not forget to **write your names and student id's at the beginning of the scripts.**

7 Cheating

We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.

Cheating Policy: Students/Groups may discuss the concepts among themselves or with the instructor or the asistants. However, when it comes to doing the actual work, it must be done by the student/group alone. As soon as you start to write your solution or type it, you should work alone. In other words, if you are copying text directly from someone else - whether copying files or typing from someone else's nots or typing while they dictate - then you are cheating (committing plagiarism, to be more exact). This is true regardless of whether the source is a classmate, a former student, a website, a program listing found in the thrash, or whatever. Furthermore, plagiarism even on a small part of the program is cheating. Also, starting out with code that you did not write, and modifying it to look like your own is cheating. Aiding someone else's cheating also constitutes cheating. Leaving your program in plain sight or leaving your computer without logging out, thereby leaving your programs open to copying, may constitute cheating depending upon the circumstances. Consequently, you should always take care to prevent others from copying your programs, as it certainly leaves you open to accusations of cheating. We have automated tools to determine cheating. Both parties involved in cheating will be subject to disciplinary action. [Adapted from http://www.seas.upenn.edu/cis330/main.html]