INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 1 - BASIC IMAGE OPERATIONS

Submission Date: 16.11.2022

Introduction

In this assignment we will learn the basics of image manipulation using histograms and filters. Although MATLAB has many built in functions, our goal is to learn image processing through doing things ourselves. Please avoid using built-in MATLAB functions unless clearly stated otherwise. However, feel free to compare your own functions to the built-in functions.
Before you start, please read the submission instructions at the end of the exercise and follow them during your work. For any question regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email schorya@post.bgu.ac.il

1 Histogram Manipulation (64 points)

In this section you will play with image histograms and quantizations.

1.1 Reading the Image (5 points)

1. Read the image named *picasso.jpg* and transform it into a gray scale image of type double using *double(rgb2gray())* function.
2. Display the image using the *imagesc()* function, determine the colormap to grayscale using *colormap()* and add a colorbar.
3. Write your own function named *dip_GN_imread(file_name)* that will return normalized gray scale image. Read the image using *imread()* function, transform it into a gray scale of type double using *double(rgb2gray())* function. Normalize the image between $[0, 1]$ using

$$\frac{Img - min(Img(:))}{max(Img(:)) - min(Img(:))} \tag{1}$$

We will use this function from section 1.3

1.2 Histogram Construction (14 points)

Use the above image named *picasso.jpg* with *double(rgb2gray())* for the following sections:

1. Write your own function named *dip_histogram(img,nbins)* that will return the histogram of the image '*img*' using '*nbins*' bins.
2. Display the generated histogram using 256 bins. Compare your result to MATLAB *imhist()* function (use a quantitative measurement).
3. We can say that the histogram roughly represents a mixture of Gaussains (Normal Distribution). To which image region (hair, face, background etc.) does each Gaussian in the histogram approximately correspond?
4. Display the histograms using 128,32,4 bins. Compare your results to MATLAB *imhist()* function (use a quantitative measurement).
5. Briefly explain the results.

Note: Here, you can use the *imhist()* function only for checking your answer.

1.3 Brightness (7 points)

From now on, use the normalized gray scale image version of *picasso.jpg* using the *dip_GN_imread(file_name)* function.

1. Write your own function named *adjust_brightness(img,action,parameter)* in which '*action*' could get either *'mul'* for multiplication or '*add*' for addition. Adjust the brightness of *'img'* using the *'parameter'*. The output of the function will be the modified image. The output of the function will be the modified image. Make sure the output image stay in the [0, 1] range.
2. Display the original gray scale image together with **four** adjusted images of increased and decreased brightness.

1.4 Contrast (11 points)

1. Write your own function named *adjust_contrast(img,range_low,range_high)* that will change the contrast of the image '*img*' and in which the *range_low,range_high* parameters will determine the new dynamic range of modified image. The output of the function will be the modified image. You should use linear mapping.

2. Calculate the modified image for a new dynamic ranges of $[0.45, 0.9]$,$[0.4, 0.5]$ and $[1, 0]$ and display the images and corresponding histograms. Explain the effect of each new range.

3. Use non-linear mapping to create modified images for a new ranges of $[0.45, 0.9]$ and $[0.2, 0.8]$. (you should clip the values that are not in the range.) Display the images and corresponding histograms. Explain the effect of each new range.

1.5 Quantization (4 points)

Quantize the original gray scale image using 6bit, 4bit, 2bit and 1bit. Brifely explain the results.

1.6 Histogram Equalization (10 points)

1. Read the image named *dog.jpg* ,transform it into a gray scale image of type double and normalize it between $[0, 1]$ using *dip_ GN_ imread(file_ name)*.

2. Use the MATLAB function *histeq()* to apply the histogram equalization on the image .

3. Display the new image and the corresponding histogram.

4. Why histogram equalization fail in enhance the image? Suggest a way to fix it.

1.7 Histogram Matching (13 points)

1. Take an image using your camera/phone/computer,read the image and transform it into a gray scale image of type double and normalize it between $[0, 1]$ using *dip_ GN_ imread(file_ name)* function.

2. Read the image named *city.jpg* and transform it into a gray scale image of type double and normalize it between $[0, 1]$ using *dip_ GN_ imread(file_ name)*.

3. Read the image named *face.jpg* ,cast it into a double type and normalize it between $[0, 1]$.

4. Display all the three images and their corresponding histograms.

5. Use the MATLAB function *imhistmatch()* to match the histogram of your image to the histogram of *face.jpg* and *city.jpg*.

6. Display the new images and their corresponding histograms.

7. Explain the results. In your explanation, consider the quality of the new images.

2 Spatial Filters and Noise (36 points)

In this section you will examine the effect of different filters on a given image.

2.1 Read the Image

Read the image named *dog.jpg* and transform it into a gray scale normalized image in the range $[0, 1]$ using *dip_GN_imread(file_name)*. Use this image from now on.

2.2 Mean vs Median Filter (11 points)

1. Write a function named *mean_filter(img,k)* that will apply a 2-D *k-by-k* mean filter on the image '*img*'. Make sure that the size of the output image is the same as the input image. Find a method to deal with the boundaries.

2. Write a function named *median_filter(img,k)* that will apply a 2-D *k-by-k* median filter on the image '*img*'. Make sure that the size of the output image is the same as the input image. Find a method to deal with the boundaries.

3. Filter the *dog.jpg* image using functions above for k=3, 5, 9, display the results and Briefly explain the results (refer to median vs. meanand the kernel size efect).

2.3 Gaussian Filter (8 points)

1. Write a function named *dip_gaussian_filter(img, k, sigma)* that will apply a 2-D *k-by-k* Gaussian filter on the image '*img*'. The smoothing kernel should be with covariance matrix of $\begin{bmatrix} sigma & 0 \\ 0 & sigma \end{bmatrix}$.
   Hint: use *meshgrid()* function in MATLAB to create the grid and apply the Gaussian formula on the grid to create your kernel.

2. Display the filtered images using *(k, sigma) = (3, 0.2), (3, 1.7), (9, 0.2), (9, 1.7)* . Briefly explain your results.

3. Subtract the original image from the filtered image. Display the new image using *'imshow(out_img, [])*'. Explain what you see.

2.4 Anisotropic Diffusion Filter (5 points)

The attached *anisodiff2D.m* function performs conventional anisotropic diffusion (Perona & Malik) upon a gray scale image. The filter aiming at reducing image noise without removing significant parts of the image content.

If you are using MATLAB version 2018a or later, use the MATLB function *imdiffusefilt()* to perform the anisotropic diffusion filter in the next sections.

1. Apply this filter over the given image.
2. Display the image and explain what you see.

2.5 Noise Filtering (12 points)

1. Create 3 new images by adding 3 different kinds of noises to the original image using *imnoise()* function. The noises are: *'salt & pepper'*, *'gaussian'* and *'speckle'*.
2. Apply the implemented filters (and the anisotropic diffusion filter) on each of the noisy images. Use kernel sizes of 3x3 and 9x9 for mean, median and gaussian filters.
3. Display the noisy and the filtered images.
4. Read the image *'square.jpg'*, cast it into a double type and normalize it between $[0, 1]$. Repeat 1-3 for *'square.jpg'*. Plot a graph which will help you understand better the effect of each filter(Hint: lecture slides)
5. Briefly explain your results. What is the effect of each filter on each noise? Which is the best filter for any given noise? what kernel size work the best? What are the pros and cons of each filter?

3 Bonus Question

Choose a noisy image (you can "trash" a clean image or start with noisy one) as you wish and try to fix it using the histogram manipulation and filters. You are more then welcome to use other ways. Display the initial image together with the modified image in the document. **Be creative! One of the images will be chosen by the course staff and it's authors will receive one bonus point to the final grade.** The staff will judge by the visual result, originality and the code.

Submission Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

The assignment is to be done in MATLAB and submitted to the course moodle page in the form of a *.zip (**not RAR**) containing ***Ex1.m*** -the main MATLAB file, other *.m files and images along with a report in the form of a PDF

file (NOT .doc). **Both the PDF and ZIP file names should be the initials and ID of both of the team members ex. 'TB-1234567_RS-7654321.pdf' and 'TB-1234567_RS-7654321.zip', respectively.**

Academic integrity: the originality of the submitted exercises **will be checked.**

Document Instructions

- Only one of the team members should submit the file
- The report should be written in Hebrew or English.
- Each section should have the relevant title as is in this document.
- Every image should be accompanied with the relevant explanation.
- The displayed images should be large enough for us to see them.
- The document should be organized and readable.

Code Instructions

- Use MATLAB version 2014b or later. If you don't have one on your computer, you can work from the computer laboratories in building 33 using VPN.
- A **main** function should call all the section functions in the correct order and should be named ***Ex1.m***.
- The first line of ***Ex1.m*** should print the full names and IDs of all team members. Use MATLAB's *disp()* function.
- Write modular functions for the subsections and reuse those functions throughout your code whenever possible.
- Every **.m* file should start with a comment containing the full names and IDs of all team members.
- Use meaningful names for all functions and variables.
- Try to avoid overriding variables.
- Write comments for every line of code that is not completely self explanatory.
- For every image displayed give a meaningful title using MATLAB's title() function.
- Use subplots whenever possible.

- All paths to files should be relative paths. If you are using subfolders use MATLAB's *fullfile()* function to construct the path to the file. Do not hard code '/' or '\' in the paths.
- The code should run completely without errors. A project with errors **will not be checked!**