INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 1 - BASIC IMAGE OPERATIONS

Submission Date: 05.11.18

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 hgazit@post.bgu.ac.il, davidit@post.bgu.ac.il, royshau@post.bgu.ac.il or arbellea@post.bgu.ac.il.

1 Histogram Manipulation

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

1.1 Reading the Image

Read the image named *picasso.jpg* and transform it into a gray scale image of type double using *double(rgb2gray())* function. Normalize the image between $[0, 1]$. This should generally be done every time an image is loaded and from now on will not be noted. Always display images using the *imshow()* function

1.2 Histogram Construction

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.

3. We can say that the histogram roughly represent 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

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

1.3 Brightness

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.

2. Display the original gray scale image together with the adjusted images of increased and decreased brightness.

1.4 Contrast

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.

2. Calculate the modified image for a new dynamic ranges of $[0.3, 0.8]$,$[0.45, 0.55]$ and $[1, 0]$ and display the images and corresponding histograms. Explain the effect of each new range.

1.5 Quantization

The default MATLAB setting *imhist()* function uses 8bit quantization (256 bins). Use the histograms to display the original gray scale image using 6bit, 4bit, 2bit and 1bit quantizations.

1.6 Histogram Matching

1. Take an image using your camera/phone/computer and transform it into a gray scale image of type double using *double(rgb2gray())* function. Normalize the image between $[0, 1]$.

2. Read by yourself about one of the algorithms for histogram matching between two images and write a short summery about the algorithm you read.

3. Use the MATLAB function *imhistmatch()* to match the histogram of your image to the histogram of your previously processed *picasso.jpg* image.

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

2 Spatial Filters and Noise

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*.

2.2 Mean vs Median Filter

1. Write a function named *mean_filter(img)* that will apply a mean of size 3x3 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)* that will apply a median filter of size 3x3 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. Display the filtered images.

2.3 Gaussian Filter

1. Write a function named *dip_gaussian_filter(img)* that will apply a Gaussian filter of size 3x3 on the image '*img*'. The smoothing kernel should be with covariance matrix of $\begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$.
   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.
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

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.

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

2.5 Noise Filtering

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.
3. Display the noisy and the filtered.
4. What is the effect of each filter on each noise? Which is the best filter for any given noise?
5. What are the pros and cons of each filter?

2.6 Larger Kernels

Repeat sections 2.2, 2.3 and 2.5. With filter size of 9x9. Explain the difference.

3 Bonus Question

Choose an image as you wish and "trash" it in the coolest way you can. For example, see the checkerboard example explained in class.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 assignment is to be done in MATLAB and submitted to the course moodle page in the form of a *.zip (**not RAR**) containing *.m files and images along with a report in the form of a PDF file (**not .doc**). The file name should be the initials and ID of both of the team members ex. `'HG-1234567_AA-7654321.pdf'`. The use of built-in MATLAB image processing functions is strictly forbidden unless the instructions says differently.

Document 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.

- Each section should have the relevant title as is in this document.

- Every explanation should be accompanied with the relevant image and every image should be explained in the text.
- The displayed images should be large enough for us to see them.
- The document should be organized and readable.

Code 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.

- 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.
- 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.
- The code should be clearly written with correct indentations (you could use automatic indentation Ctrl+A followed by Ctrl+I).
- 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!**