EEE 391
Basics of Signals and Systems
Spring 2019–2020
MATLAB Assignment 2
Due: 14 April 2020, Tuesday by 23:55 on Moodle

Part I: RGB Components

In the first part of the assignment, you need to work with one of the four images (image0.jpg, image1.jpg, image2.jpg, image3.jpg) provided on Moodle. First, determine the remainder of your ID number when divided by 4. If the remainder is i, where $i = 0, 1, 2, 3$, you will work with image$i$.jpg.

Let n and m be the independent variables (integer indices) indexing the pixels (matrix elements) along the horizontal (x) and vertical (y) directions of the image, respectively.

Use the command `A=imread('nameofimage.jpg')` to read the image file 'nameofimage.jpg' and convert it to a matrix A. Here, A is a 3-dimensional (3-D) matrix such that $A(m, n, :)$ represents the RGB components of the pixel located at (m, n). $A(m, n, 1)$, $A(m, n, 2)$, and $A(m, n, 3)$ correspond to R, G, and B values of the spatial location (m, n), respectively, and each can take integer values between 0 and 255.

(a) Find all spatial locations (m, n) of A with R > 140. Create a new binary matrix (image) that has 1 in the corresponding spatial locations, otherwise 0. Display the binary image and comment on the result. Display the images as 2D images, not as 2D or 3D graphs/plots of values. To display an image provided as a matrix `Y`, use the command `imshow(Y)`.

(b) Repeat the same for G > 140. Display the binary image and comment on the result.

(c) Repeat the same for B > 140. Display the binary image and comment on the result.

(d) Repeat the same for the spatial locations having the following RGB values: R > 140, G > 140, B < 30. Display the binary image and comment on the result.

(e) Create a new image named "gray.jpg" by taking the average of the color components (R+G+B)/3 for each spatial location, that is, the image matrix will be 2-D. Display the image.

Part II: Mean Filtering

In this part, you will use "gray.jpg" that you obtained in Part I. The mean filter is one of the simplest linear filters used for processing image signals. Let I denote the image matrix and J be the mean filtered version of I. Then, each pixel of J is given by

$$J(m, n) = \frac{1}{(2N+1)^2} \sum_{i=m-N}^{m+N} \sum_{i=n-N}^{n+N} I(i, j)$$

where $M = (2N+1)^2$ denotes the total number of pixels in the neighborhood of the pixel (m, n) considered for mean filtering. The size of the neighborhood, M, controls the amount of filtering. While filtering, assume that the values lying outside of the image are zero.

(a) Write a function to perform mean filtering without using any available function in MATLAB including `imfilter`$(\cdot)$. This function should take any image matrix I and M as inputs and provide the mean filtered version of I at its output.

(b) Corrupt the gray image "gray.jpg" by adding zero-mean Gaussian noise with variance 64 to the image. You can use the following commands for this purpose:

```
gaussnoise = 8*randn(size(imgray,1), size(imgray,2));
imgaussnoise = uint8(double(imgray) + gaussnoise);
```

Here, `imgray` denotes the image matrix for gray.jpg. Display the corrupted image.

(c) Filter the `imgaussnoise` matrix with the function you implemented in (a), for $M = 9, M = 25$, and $M = 121$. Display the filtered versions. Comment on the results.

(d) Change the variance of the Gaussian noise to 1024. Then, reconstruct `imgaussnoise` matrix. Repeat the same operations as in (c).

(e) In this part, change the type of noise and use "salt and pepper" type noise. You can use the following commands for this purpose:

```
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
imsaltnoise( find( noisypixels ≤ ( 1 / 16 ) ) ) = 255;
imsaltnoise( find( noisypixels ≥ ( 15 / 16 ) ) ) = 0;
```

Try to understand this code. How does the noise corrupt the image matrix? Explain. Display `imsaltnoise` matrix, as well.

(f) Then, for $M = 9$, $M = 25$, and $M = 121$, obtain the mean filtered versions of the `imsaltnoise`. Display them. Comment on the results.

Comment on:

(i) How can you describe the visual effect created by the filter?

(ii) What happens to the details in the image? How is this affected by changing the value of M?

(iii) Compare the visual effect in the dimension n versus the dimension m (horizontal versus vertical)? Is there any difference?

(iv) Comment on what happens close to the edges of the image? How is this affected by changing the value of M?

Part III: Median Filtering

In this part, you will again use "gray.jpg" from Part I. Let I denote the image matrix and J be the median filtered version of I. Then, each pixel of J is given by

$$J(m,n) = \text{median}\{I(i,j)\}_{i=m-N,j=n-N}^{i=m+N,j=n+N}.$$

This time, instead of averaging the pixel values in the neighborhood of the pixel (m,n), we take their median. Again, assume that the values lying outside of the image are zero. Write a function for median filtering without using any available function in MATLAB including `imfilter`$(\cdot)$ and `medfilt2`. This function should take any image matrix I and the parameter $M(=(2N+1)^2)$ as inputs and provide the median filtered version of I at its output.

Then, repeat the steps (b)–(f) in Part II for the median filter. Comment on the results addressing the same questions as in Part II.

Part IV: Edge Detection with the Sobel Operator

In this part, you will work with one of the four images (image0part4.jpg, image1part4.jpg, image2part4.jpg, image3part4.jpg) provided on Moodle based on the remainder of your ID divided by 4. For example, if the remainder is 3, you will work with image3part4.jpg.

Let I be the image matrix of interest. Define two matrices as:

$$G_x \triangleq \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \qquad \text{and} \qquad G_y \triangleq \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

where $*$ denotes the 2-D convolution operator. Note that G_x and G_y aim to detect edges in the x and y directions of the image, respectively. The output of the Sobel operator when applied to the image I is given by $G = \sqrt{G_x^2 + G_y^2}$.

In order to find G, you can use `conv2`$(\cdot)$ command of MATLAB. Note that I is a 3-D matrix, hence, in order to compute G_x and G_y, you need to perform the convolution operation for each one of the RGB components. Display both I and G. Comment on the results.

Note: In this assignment, to modify the styles of the plots, to add labels, and to scale the plots, use only MATLAB commands; do *not* use the GUI of the figure windows. When your program is executed, the figures must appear exactly the same as you provide in your solution. You need to write your MATLAB codes not only correctly but efficiently as well.

Submit the results of your own work in the form of a well-documented report on Moodle. Borrowing full or partial code from your peers or elsewhere is not allowed and will be punished. Please include all evidence (plots, screen dumps, MATLAB codes, MATLAB command window print-outs, etc.) as needed in your report. Append your MATLAB code at the end of your assignment, do not upload it separately. The axes of all plots should be scaled and labeled. Typing your report instead of handwriting some parts will be better. Please do not upload any photos/images of your report. Your complete report should be uploaded on Moodle as a single good-quality pdf file by the given deadline. Please try to upload several hours before the deadline to avoid last minute problems that may cause you to miss the deadline. Please DO NOT submit files in other formats or any files by e-mail.