EE 605: Digital Image Processing - Assignment 2

Teacher: Prof. Nitin Khanna (nitinkhanna@iitgn.ac.in)

Guidelines:

- You can do this assignment in Matlab and/or Python.
- There should be only one .zip file (containing all your .m files, figures/images corresponding to results and a .pdf and corresponding .tex or .doc(x) file for the summary report). Filename shall be in the format as follows: DIP-A2-rollno.zip (For e.g. DIP-A2-16110145.zip). Submit this zip file on Google classroom.
- There shall be one report for the complete assignment 2, as .tex or .doc(x) file (with corresponding .pdf file) showing the input image, output image, discussion about results etc. corresponding to different questions. Also include the description of parameters chosen and any specific observations made. Filename for the report shall again be in format as follows: DIP-A2-report-rollno.tex or or DIP-A2-report-rollno.doc(x) and DIP-A2-report-rollno.pdf.
- Questions for which the output could be computed using inbuilt functions in Matlab, mention that function, show the corresponding output. Provide clear discussion about any differences observed. Compare the time taken by your function with the time taken by corresponding inbuilt function.
- Include a readme.txt file mentioning different steps for running your program and any other comments from your side.
- Make suitable assumptions as needed and clearly mention them at appropriate places in your code, readme file and report.
- In writing code for the following problems, you can use Matlab's built in function for image reading and writing (imread and imwrite), apart from these, please do not use any other function from Matlab's toolboxes related to image processing (except the functions mentioned as allowed function under a particular question). Write the code from scratch, using basic constructs like loops etc. Please confirm with me by posting a query in Google classroom if you think any other in-built function is unavoidable to use. *Note: Allowed in-built Matlab functions/constructs: same as that in Assignment-1.*
- Matlab code should be very well documented. The evaluation will also give consideration to your writing efficient code/algorithms, using dynamic and not-static declarations etc., code having proper indentation and self-explanatory comments etc.
- Unless you state otherwise, it is implicitly assumed that you wrote the code yourself without any help from your friends/online-resources. If you take some help, you must mention that explicitly in your report as well as in your Matlab code comments. Plagiarism related polices as per Institute rules will be applicable and strictly enforced.

Note: Images given in the questions are just shown for reference purpose, they are not upto scale. Feel free to let me know if you need any further clarification.

1. (Adjust image intensity values or colormap) In this question, you need to carefully go through an inbuilt function in Matlab and then write your own function which will have exactly same syntax as the inbuilt function and perform exact same operation as this inbuilt function. Then, you need to compare the execution of these two function for two or more choices of input parameters. Explain any difference in output between these two functions, if any.

You need to write your own function using only those inbuilt function which were allowed in Assignment-1 and basic constructs such as loops etc., and no other inbuilt function, check with the teaching team if you believe that use of a particular inbuilt function is absolutely essential.

Write your own function imadjust_rollno(), which should be equivalent to Matlab's inbuilt function imadjust, under following variety of syntax (in function name, replace rollno with your own roll number).

- J = imadjust(I)
- J = imadjust(I,[low_in; high_in],[low_out; high_out])
- J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)
- newmap = imadjust(map,[low_in; high_in],[low_out; high_out],gamma)
- RGB2 = imadjust(RGB1,...)

2. (Histogram Equalization and Matching)

   In this question, you need to carefully go through an inbuilt function in Matlab and then write your own function which will have exactly same syntax as the inbuilt function and perform exact same operation as this inbuilt function. Then, you need to compare the execution of these two function for two or more choices of input parameters. Explain any difference in output between these two functions, if any.

   You need to write your own function using only those inbuilt function which were allowed in Assignment-1 and basic constructs such as loops etc., and no other inbuilt function, check with the teaching team if you believe that use of a particular inbuilt function is absolutely essential.

   Write your own function histeq_rollno(), which should be equivalent to Matlab's inbuilt function histeq, under following variety of syntax (in function name, replace rollno with your own roll number). You are not allowed to use imhist or hist or equivalent functions.

   - $[J, T] = histeq(I)$
   - J = histeq(I,n)
   - J = histeq(I,hgram)
   - (Addition to Matlab) J = histeq(I,I1,verbose_fig) This syntax should provide an extra functionality of performing histogram matching, when both I and I1 are two dimensional matrices (gray-scale images). Point transformation should be performed on I to generate output J, such that the histogram of the output image J should be as close as possible to the histogram of the input I1. If the input verbose_fig = 1, the function should also display histograms of I, I1 and J with appropriate labels and title.

3. (Spatial Filtering)

   Write your own function imfilter_rollno(), which should be similar to Matlab's inbuilt function imfilter (there are differences in syntax but purpose is similar),(in function name, replace rollno with your own roll number). Your function should have following syntax, with only f and w as mandatory input arguments.

   g = imfilter_rollno(f , w, filtering_mode, boundary_options , size_options)

   where f is the input image, w is the filter mask, g is the filtered result, and the other parameters are summarized below. The filtering_mode is specified as ' corr ' for correlation (this is the default) or as ' conv ' for convolution. The boundary_options deal with the border-padding issue, with the size of the border being determined by the size of the filter. The size_options are either ' same ' or ' full ' ('same' is the default).

4. (Thresholding)

   In this question, you need to carefully go through an inbuilt function in Matlab and then write your own function which will have exactly same syntax as the inbuilt function and perform exact same operation as this inbuilt function. Then, you need to compare the execution of these two function for two or more choices of input parameters. Explain any difference in output between these two functions, if any.

   You need to write your own function using only those inbuilt function which were allowed in Assignment-1 and basic constructs such as loops etc., and no other inbuilt function, check with the teaching team if you believe that use of a particular inbuilt function is absolutely essential.

   Write your own function imquantize_rollno(), which should be equivalent to Matlab's inbuilt function imquantize, under following variety of syntax (in function name, replace rollno with your own roll number).

Options	Description
Filtering Mode	
`'corr'`	Filtering is done using correlation
`'conv'`	Filtering is done using convolution
Boundary Options	
`P`	The boundaries of the input image are extended by padding with a value, `P` (written without quotes). This is the default, with value 0.
`'replicate'`	The size of the image is extended by replicating the values in its outer border.
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its border.
`'circular'`	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
`'full'`	The output is of the same size as the extended (padded) image
`'same'`	The output is of the same size as the input. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image

Figure 1: Imfilter Options

- quant_A = imquantize(A,levels)
- quant_A = imquantize(___,values)
- $[quant\_A, index] = imquantize(\_\_\_)$

To find out the levels input in above function, you also need to write your own function multithresh_rollno(), which should be equivalent to Matlab's inbuilt function multithresh, under following variety of syntax (in function name, replace rollno with your own roll number).

- thresh = multithresh(A)
- thresh = multithresh(A,N)
- (Addition to Matlab) thresh = multithresh(A,method) This syntax should provide an extra functionality of finding a single threshold using basic global thresholding method discussed in the class. Input method can either be "Ostu" (this is default) or "Basic Global".

5. (Huffman Coding and Decoding) Write a function and corresponding test script for performing Huffman Coding (use the conventions as described in the class). Input to the function should be: 1) either a matrix with two columns (first column is symbol number and second column it's probability and each row indicating a symbol) or a single channel image, and 2) (optional argument) a 1-D array listing symbols numbers in the message to be coded (if this argument is missing then the function should return the result of applying Huffman Coding on the input single channel image.)

   Output of the function should be the result of applying Huffman Coding on the specified data, specified as a string of '0' and '1' and the probability table used for encoding.

   Also, write a function with appropriate inputs and output which will perform decoding of the message coded by the above function.

Your test script should apply this on a sample image and save the output as a .mat file. Perform decoding of this message and save the resulting decoded message as an image.

6. (Arithmetic Coding) Write a function and corresponding test script for performing Arithmetic Coding. Input to the function should be: 1) either a matrix with two columns (first column is symbol number and second column it's probability and each row indicating a symbol) or a single channel image, and 2) a number 'N' which indicates the number of symbols which are to be coded together (if the total length of the message to be coded is not a multiple of N, then last message sequence will contain fewer than N symbols), and 3) a flag indicating whether the output should be binary fractions or decimal fractions (in either case the function should give the binary/decimal number with least number of digits needed to represent a sequence of symbols), and 4) (optional argument) a 1-D array listing symbols numbers in the message to be coded (if this argument is missing then the function should return the result of applying Arithmetic Coding on the input single channel image.)

   Output of the function should be the result of applying Arithmetic Coding on the specified data, specified as a column vector of appropriate size and the probability table used for encoding.

   Your test script should apply this on a sample image and save the output as a .mat file.

7. (Arithmetic Decoding) Write a function and corresponding test script for performing Arithmetic Decoding. Input to the function should be: 1) probability table, matrix with two columns (first column is symbol number and second column it's probability and each row indicating a symbol), and 2) size of the complete original message, ex. either [20000 1] or [100 200], and 3) a number 'N' which indicates the number of symbols which were coded together (if the total length of the message to be coded is not a multiple of N, then last message sequence will contain fewer than N symbols), and 3) a flag indicating whether the encoding is done as binary fractions or decimal fractions, and 4) a column vector corresponding to the encoded data

   Output of the function should be the result of applying Arithmetic Decoding on the specified data, specified as a column vector of appropriate size or a 2-D matrix of appropriate size.

   Your test script should read the encoded message from a .mat file and save the resulting output as a single channel image.

8. (Run-length Coding)

   Write a function and corresponding test script for performing Run-length Coding. Input to the function should be: 1) a matrix representing a single channel image, and 2) a flag indicating whether the encoding is to be done directly on grayscale image, or it's binary planes or it's grayscale planes

   Output should be a string representing the encoded data in a form similar to (g_i, n_i).

   You should also write a function which will generate a test image similar to that shown in Figure 2 (input to this function should be three numbers specifying radius of the center circle, sides of the next two squares and three sets representing range of intensities from which the pixel values are chosen for each of these three shapes). Finally, write a test script which will apply run length coding on a test image generated by your above function.

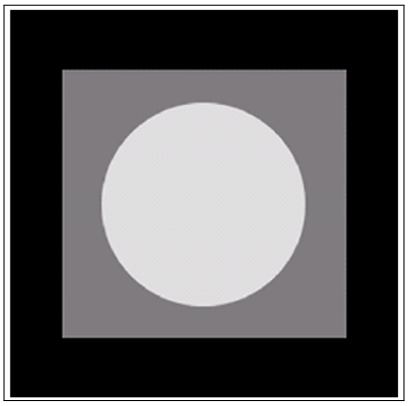

Figure 2: Sample Test Image