Biomedical Imaging Informatics Practical #2

Instructions:

This group assignment is expected to **require 5-8 hours to complete.** Please plan accordingly.

Due Date: **3pm EST, Oct. 17th (Thursday)**.
Submission Files: the **signed HONOR CODE page** (see HONOR CODE page), and **one zipped program folder file** (i.e., containing all code files)

Submission Protocol:
Upload the zipped file to your home directory on knn.bme.gatech.edu, and then send an email to 8813grading@lists.gatech.edu to inform us that you have submitted your assignment. For your email subject line, please use: BMED8813 last name 1, last name 2, group #, PRACTICAL #2.

Submission File Protocol:
(1) Your submission will be one zipped folder file. The folder should contain all of the necessary files (i.e., the main program file, possible additional function files, and necessary test data files) to run your code and show the results.
(2) The folder should be named: < last name 1>_<last name 2>_..._<group #>
(3) In your folder, code files should be named as: <last name 1>_<last name 2>_..._<group #>_<code name>.m (or .pl)
(4) Please write well-commented code, and include any explanations you think will help the instructor understand your program. If your program does not work completely, comments might help you get partial credit. However, you will get **NO CREDIT** if your program generates errors, generates warnings, or outputs "junk".

If you have problems, please contact the instructors at 8813grading@lists.gatech.edu with the proper email subject line as instructed in the syllabus.

HONOR CODE

The conditions of this assignment are subject to the Georgia Institute of Technology Academic Honor Code.

I pledge that the work in this assignment, including all written code, represents the original work of BMED8813 BHI Group ____ [write group #]. I have NOT communicated with anyone (other than my group members) about the contents of this assignment, nor participated in or observed any conduct prohibited by the Honor Code.

Student Signature _____

Student Signature _____

Student Signature _____

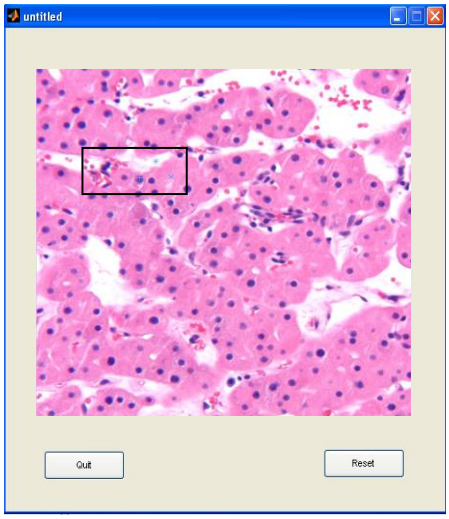
NOTE: Please print out this page, sign the page, scan the signed page, and submit with your programming code.

If you have a problem finding a scanner to scan in the page, please type in the above HONOR CODE statement in your email message body and sign with your name.

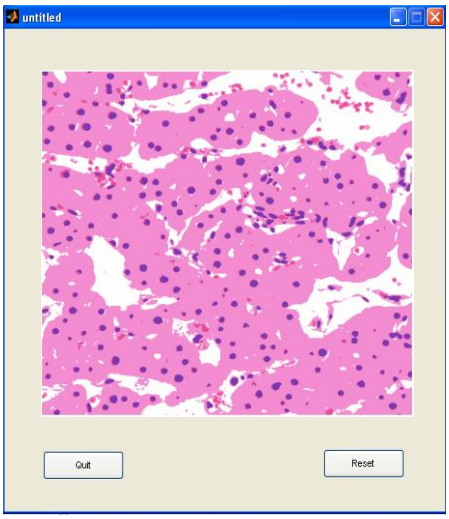
Any assignment without the HONOR CODE PAGE will be void to ZERO grade by default.

1) Develop a user-interactive color segmentation method for Hematoxylin and Eosin (H&E) stained histological images. H&E staining of a histological image enhances four colors: blue-purple, white, pink and red. These colors correspond to specific cellular structures. Blue-purple, white, pink, and red correspond to nuclei, no-stain, cytoplasm, and red-blood cells, respectively. (40 pts)

a) Create a GUI in MATLAB containing a figure axis for image display and two buttons at the bottom. The first button is "Reset". The functionality of Reset is to restore the application to the initial state (for testing). The second button is "Quit". The functionality of "Quit" is to end the program and close the figure. Display the sample image ("Test_Image.mat") on the GUI and ask the user to select a sample for each color in the order blue-purple, white, pink and red (Hint: you can use "impixel" function).



GUI for user-interactive segmentation. The 'x' marks indicate user-selected samples.



Pseudo-colored color segmentation results.

b) The samples selected above represent centers of the respective color cluster or group. Now, please divide all pixels into these color clusters based on their distance to the centers. Compute the Euclidian distance of all pixels in the image to each of the four centers in the RGB space (Hint: you can use the "pdist2" function). Then, assign each pixel to one of the four groups such that the center of the selected group is the shortest distance to the pixel.

c) Please create a pseudo colored RGB image representing the segmentation result. Represent each pixel with the RGB color of its cluster's center. As shown above.

d) Please convert RGB into HSI for the images, and show the color distribution in the 2-D HS space.

2) Compare the discrete cosine transform (DCT) and the discrete Fourier transform (DFT). The purpose of this comparison is to determine which transform can reduce the total number of bytes required for data storage. (Hint: for the "zero-valued" coefficients in the transform domain, you only need 1 bit for each pixel.)(40 pts)

a) Use the provided image "Test_Image_512.mat" to perform two dimensional DCT ("dct2" in Matlab). The original image is of size 512x512 pixels.

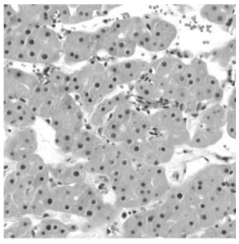
b) Reduce the DCT image size to (128x128) which is 16 times smaller than the original image, preserving the most informative portion. Apply the inverse DCT transform to the 128x128 DCT image with size expansion to 512x512 ("idct2" in Matlab) to recover the original image. Please show the steps:
   i) the starting grayscale image,
   ii) the DCT,
   iii) the final inverse DCT image, and
   iv) the difference residual image (difference between starting and final image), as shown in the figure below.

c) Repeat steps (i) and (iv) in part (b) with the discrete Fourier transform ("fft2" and "ifft2" in Matlab).

d) What differences do you observe in the results from DCT and DFT?

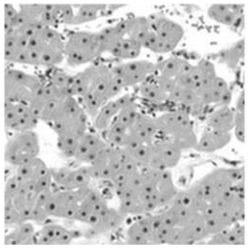
e) Which method would you prefer to use for reducing the total number of bytes required for data storage? Try to justify.

(NOTE: you MUST clearly label the results and explain why the final images are not exactly the same as the original image.)

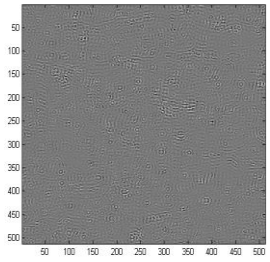


Grayscale 1024x1024

DCT



Inverse DCT Image



Difference Image

3) In class, a few image interpolation techniques were suggested for image zooming. In this question, please try the interpolation techniques to increase the resolution of histological images. We have provided a benchmarking image in various resolutions— 1024x1024, 512x512, 256x256, 128x128 and 64x64. (35 pts)

a) Use the "interp2" function in MATLAB and implement a few image interpolation schemes: 1) "Nearest neighbor interpolation", 2) "Linear interpolation", and 3) "Cubic interpolation".

b) Convert the 512x512, 256x256, and 128x128 resolution images to a 1024x1024 resolution image using interpolation. Then, calculate the residual image between the original 1024x1024 benchmark image and all other interpolated images of the same size. Please compare the results visually, and describe what you observe.

c) Compute the root mean square error (RMSE) between the interpolated images and the original benchmark 1024x1024 image given by

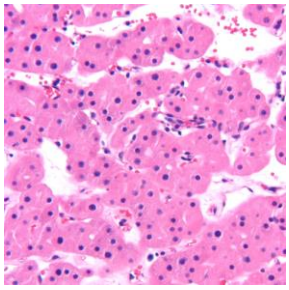
$$e_{rms} = \left[\frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \left[\left(I_R(x,y) - I_R'(x,y)\right)^2 + \left(I_G(x,y) - I_G'(x,y)\right)^2 + \left(I_B(x,y) - I_B'(x,y)\right)^2 \right] \right]^{1/2}$$

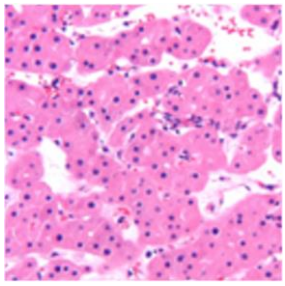
NOTE that *I(x,y)* and *I'(x,y)* represent *M*x*N* dimensional original and interpolated images, respectively. $I_C(x,y)$ represents the C-channel intensity of the pixel at (*x*,*y*) location, where $C \in (R, G, B)$.

Plot a graph of the RMSE (i.e., averaged difference between the interpolated image and the original benchmark image) versus the different sizes of interpolation for all three interpolation methods (shown below).

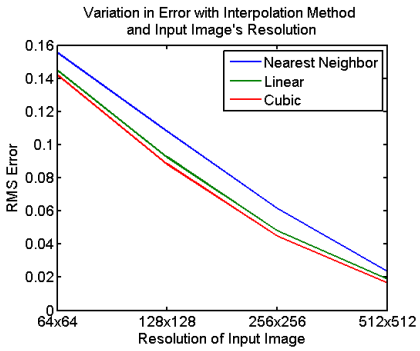
d) Smooth the interpolated images using a Gaussian filter (Hint: you can find it in the MATLAB function library), and recalculate the RMS error graph.

Original 1024x1024 resolution image





1024x1024 interpolated image from 128x128 input image using "Cubic interpolation"



Plot of RMS error of 1024x1024 interpolated images generated from different original benchmark image sizes, using three interpolation methods.

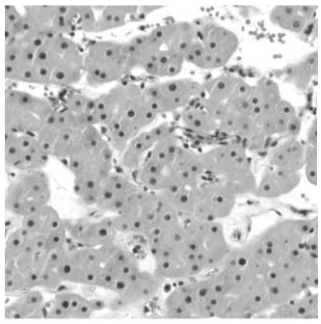
4) Compare various edge detection methods for histological images. Also, compare the performance of interpolation schemes in the above Problem Part 3 using edge detection. (35 pts)

a) Please extract edges in the 1024x1024 resolution original benchmark image **(Hint: please convert to grayscale first)** using the following edge detector operators (all available in MATLAB):
   i) "Sobel",
   ii) "Prewitt"
   iii) "Roberts", and
   iv) "Canny".
   Compare the performance of these edge detectors. Which edge detector works the best, and why?

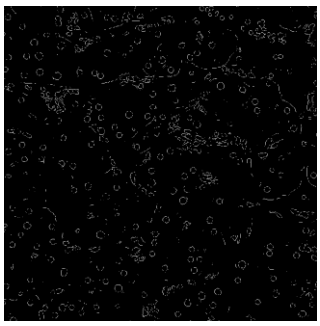
b) Please use the "Canny" operator to extract edges in the three interpolated 1024x1024 images (using NN, Linear, Cubic) generated from the 256x256 benchmark image in **Problem Part 3b**, and then compare them with the result in **Problem Part 4a.iv** above**.** Please describe what you observe about the edges extracted from the three different interpolated images *versus* the edges extracted from the original benchmark image.

c) Please use the "Canny" operator to extract edges in the Gaussian filtered three interpolated 1024x1024 images (using NN, Linear, Cubit) generated from the 256x256 benchmark image in **Problem Part 3d.** Then, please compare the results with the result in **Problem Part 4b** above, and comment on how smoothing has impacted the results.

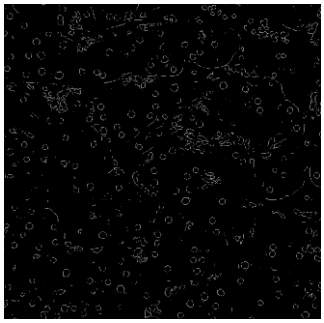
Hint: See some example processing result images below for your reference.



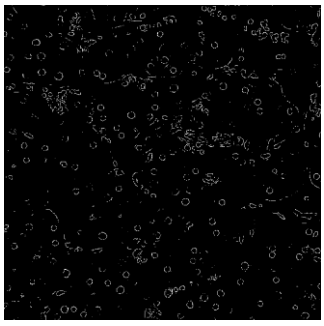
Grayscale 1024x1024 resolution Image



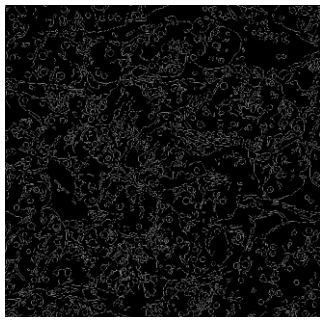
Edge detection by "Sobel"



Edge detection by "Prewitt"



Edge detection by "Roberts"



Edge detection by "Canny"