CSC 589 Mid-term exam (take-home)

# Instructions

Due: Thursday, March 19th, 11am (in Class). Late work will receive zero credits!

**Teamwork:** This is not a regular assignment. You are expected to finish on your own. Collaborations are not allowed. Everyone must turn in your own code and images. Please do not copy code from the Internet. Plagiarism will be reported to the university.

**Program language:** you can choose to do it either in Python or MATLAB. For your instructor to grade it properly across all level of students, OpenCV is not allowed in this exam and it is not necessary. It also will not simplify your life. You are welcome to check your work with existing packages and also photoshop. But you must write your own MATLAB or Python code. Recommended packages are Scipy and Numpy. MATLAB image processing toolbox.

**Write up:** Please upload a word or pdf document reporting each step of what you have done and upload your final images results.

Please write well-commented and well-read code! Please put your name and dates on top of the scripts as comments.

Please name your folder (with your full name) and zip it and upload to blackboard.

**Total points:** 60pts

1. (30pts) De-convolution

You’ve taken the most beautiful picture and are editing it. You decide to see what it might look like if it were blurred slightly. Unfortunately, your graphics package crashes just after the blurring and saves over the original. What will you do?

In class, we showed that convolving to signals is equivalent to multiplying their Fourier transforms. In this problem you will explore the limits of undoing a convolution.

For this problem, turn in the necessary code to complete the following steps. Also turn in the resulting images.

1. Make sure the image is stored as floating point values. You can use imread.misc.astype(float)

2. Blur the image using a convolution kernel. (You should use full or circular convolution to make the later parts work.) You can use a Gaussian filter.

3. Compute the Fourier Transform of the kernel. Make sure you pad the kernel so that it is the same size as the original image.

4. Using the DFT of the kernel and the DFT of the blurred image, how would you invert, or undo, the blurring operation?

5. Try this and see how well it works

6. Now, quantize the blurred image to 8 bits per pixel, (by converting it to unsigned characters, for example), then convert back to floating point values.

7. Now try inverting the convolution. How do the results compare? What do you think caused the difference? Discuss your results.

# 2. (30pts) Median Filter

In the earlier lectures, we discussed the median filter. It works like a local averaging filter, except instead of taking the mean of a window, the median filter uses the median value in the window. (Look up median in Wikipedia if you do not remember the difference between a mean and a median). MATLAB and Python both provide median filtering operations, medfilt2 and medfilt2d respectively.

Write your own median filter (mymedfilt2.py) and check your results with the medfilt2 and medfilt2d in MATLAB and Python. This would be similar as the imfilter you wrote for project 1. You also have to pad your images. For color image, filter for each channel and combine.

Filter the two images for this problem set with median and averaging filters of 3 different sizes. How would you describe the difference in output between the mean and median filter? How does using the median cause these differences?

# Rubric

For both problems:

20pts 15pts for accurate implementation, 5pts code efficiency and clarity (5pts).

10pts for write up and discuss your results

-5pts if you didn’t zip your file properly and didn’t put your full name on both the zipped and the original folder.

Code that doesn’t compile will receive 50% reduction automatically.