# EEE-6512 Image Processing and Computer Vision
# Fall 2020 Homework #4
# October 10, 2020

## Due: October 24, 2017, 11:59 PM

This assignment should be completed individually by the student. Proper citation should be provided for any references used.

**Part I Textbook Questions [50 points]**

Answer the following questions from the textbook:

- Chapter 4: 4.3, 4.8, 4.12, 4.15, 4.20 (a-c, inclusive)
- Chapter 5: 5.7, 5.23
- Chapter 6: 6.4, 6.19, 6.20

**Part II MATLAB Programming [50 points]**

Please read requirements carefully. Solutions that do not follow provided specifications will not receive credit. Allowed MATLAB toolbox functions include ones which read an image, show an image, pad an image, compute DFT and inverse DFT, apply a median filter, and time a function. No other MATLAB toolbox functions are allowed. Test your functions on *escher.png*.

- Write a function, *mySpatialFilt*, which:
  - accepts a grayscale image and filter (you may assume odd height and width size)
  - filters the image using the filter by convolving in the spatial domain (as described in the text). Handle edges by padding with mirror reflections of the array along the border.
  - shows and returns the filtered image.
- Write a function, *myFrequencyFilt*, which
  - accepts a grayscale image and filter (you may assume odd height and width size)
  - filters the image using the filter by multiplying in the frequency domain (as described in the text)
  - shows the frequency-domain DFT image before filtering
  - shows and returns the filtered image
- Using your *mySpatialFilt, myFrequencyFilt,* the MATLAB function for median filtering, and the provided *escher.png,* fill out the following table with the corresponding times to run each function. **Note: for the timing part, turn off all imshow functionalities by commenting the corresponding lines out. Uncomment them back in before turning in the assignment.**

| **Method**<br>**Filter** | **Run time when Filt size (k) = 3** | **Run time when k = 203** | **Run time when k = 403** |
|---|---|---|---|
| *mySpatialFilt*<br>Gaussian kxk kernel* | | | |
| *mySpatialFilt* | | | |

| | | | |
|---|---|---|---|
| Gaussian kernel in separable form (1xk, kx1) | | | |
| *myFrequencyFilt*<br>Gaussian kxk kernel | | | |
| *myFrequencyFilt*<br>Gaussian kernel in separable form (1xk, kx1) | | | |
| MATLAB toolbox method for median filtering<br>Median kxk filter | | | |

- Programming questions (PUT IN WRITEUP):
    - Include the filtered images from the following runs:
        - *mySpatialFilt*, Gaussian 11x11 kernel
        - *mySpatialFilt*, Gaussian kernel in separable form (1x11, 11x1)
        - *myFrequencyFilt*, Gaussian 11x11 kernel ← include the DFT of the image before filtering as well
        - *myFrequencyFilt*, Gaussian kernel in separable form (1x11, 11x1) ← include the DFT of the image before filtering as well
        - MATLAB median filter, median 11x11 kernel
    - Questions:
        - What is the difference between gaussian versus median filtering, with respect to the output filtered image?
        - What is the difference between filtering in the spatial domain versus in the frequency domain, with respect to time?
        - What is the difference between filtering with a kxk filter versus the separable form of the same filter, with respect to time?
        - When filtering with a Gaussian kernel, what is the effect on the output when the sigma value is changed, but the filter size remains the same? How about when sigma value remains the same, but the filter size is changed?

* Note: for all Gaussian filters in this assignment, choose the standard deviation (sigma value) such that the filter size, k, is approximately 6*sigma rounded to an odd integer value.

To receive full credit for this assignment, you should submit three files. 1.) A document containing answers to the textbook and programming questions (.DOC, .DOCX, or PDF file) 2.) An M-file containing commented MATLAB code for the program *mySpatialFilt*. 3.) An M-file containing commented MATLAB code for the function *myFrequencyFilt*. **Reminder: ensure both code submissions show images when run, in case it is still commented out from the timing question**. Students should ensure that their M-files execute without errors to avoid receiving point deductions.

**Part III Extra Credit [25 points]**

Please read requirements carefully. Solutions that do not follow provided specifications will not receive credit. You are free to use any built-in/toolbox functions within MATLAB to accomplish this task, except functions from the deep learning toolbox. Data and background were taken from the Sign Language MNIST Kaggle dataset page [1].

**Background**: The original MNIST image dataset of handwritten digits is a popular benchmark for image-based artificial intelligence methods but researchers have renewed efforts to update it and develop drop-in replacements that are more challenging for computer vision and original for real-world applications. American Sign Language (ASL) MNIST is one such dataset, consisting of images of hand gestures which represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion). This has applications in live ASL/spoken word translation. See provided *asl_reference.png* for the ASL letters. For more information, please see [1].

**Data**: The dataset format is patterned to match closely with the classic MNIST. Data is stored in csv format with:

1. a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions)
2. pixel1,pixel2….pixel784 which represent a single image. Data was preprocessed by cropping to hands-only, gray-scaling to uint8 bit depth, and resizing to a 28x28 pixel image.
3. For this assignment, we are only concerned with the letters A-D, inclusive. Please use the provided *asl_mnist.csv*.

**Challenge**: Write a function, *myASLTranslate*, which:

- accepts a single 28x28 uint8 greyscale image and returns a single character, either “A”, “B”, “C”, or “D”.
- You must:
    - Use at least one filter on the grayscale image
    - Use at least one morphological image processing operation
    - Use at least one region property from section 4.4
    - Include in your report the accuracy of your function on all data in *asl_mnist.csv*
- Note: code will be tested on 25 randomly sampled images taken from the provided *asl_mnist.csv*. Code must achieve at least 80% accuracy on the sampled dataset to receive any credit.

To receive full credit for the extra credit part, you should submit two files. 1.) A document containing an explanation of how your code works, with enough information/intermediate images for another student in the class to roughly duplicate your work and understand why each step was taken (.DOC, .DOCX, or PDF file) 2.) An M-file containing commented MATLAB code for the program *myASLTranslate*. Students should ensure that their M-files execute without errors to avoid receiving point deductions.

**References**

[1] “Sign Language MNIST.” https://kaggle.com/datamunge/sign-language-mnist (accessed Oct. 09, 2020).