CS 6643 - Computer Vision, Spring 2020 **James Fishbaugh**

Project 1

Out: 2 / 26 / 2020
Due: 3 / 09 / 2020 (deadline: 11:55PM)

Late submissions: Late submissions result in 10% deduction for each day. The assignment will no longer be accepted 3 days after the deadline.

Office hours:

		Mon	Tue	Wed	Thur	Fri
James Fishbaugh	Office #1138	2-3PM		2-3PM		
Sherry Chou	hrc304@nyu.edu		1-2PM			2-3PM
Junqi Zhang	jz3350@nyu.edu		3-4PM		2-3PM	

Location: TA Lounge 8th floor 370 Jay

Please read the instructions carefully. Note: we will not be running code. Rather, we will check your code to make sure your implementation is your own, and it matches your results. Your grade is primarily based on your written report. This means going beyond just showing results. You should produce a standalone lab report, describing results in enough detail for someone else to follow. Please submit a single PDF/HTML with all code included as an appendix.

A) Programming Questions

The purpose of this programming project is to get familiar with the implementation and application of spatial filters for image denoising, derivatives, and template matching.

Hint for implementation: Most image processing require using float or double type for image arrays. You can map those images back to integer at the very end of the processing for display purposes, and during this operation you can also optimally scale and shift the result to [0, 255] after calculation of minimum and maximum intensities across the resulting image.

A1a) Image Denoising: Implementation

Box filter: Implement a function which creates a box filter with size *n*x*n* given by the user. Make sure the filter has an odd size, i.e. if the user gives an even *n* just increase the size by 1.

```
def create_box_filter(n):
  # Create box filter g of size n*n
  return g
```

Gaussian filter: Implement a function which creates a Gaussian filter with standard deviation σ given by the user. Make sure the filter has an odd size, i.e. if the user gives an even *n* just increase the size by 1. Note, do not hardcode values, sample the Gaussian function, e.g.

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

You may use the "3σ rule" for determining the size *n*x*n* of the filter. See the function prototype below:

```
def create_gaussian_filter(sigma):
  n = 2*math.floor(3*sigma)+1    # The size nxn of our filter

  # Create Gaussian filter g of size n*n
  return g
```

Convolution: Implement 2D convolution (denoted *) without the use of built in functions. Your function will take as input two 2D arrays, a filter *f* and an image *I,* and return $f*I$. You can handle the boundary of *I* in any reasonable way of your choice, such as padding with zeros based on the size of the filter *f*.

```
def convolution2D(f, I):
  # Handle boundary of I, e.g. pad I according to size of f

  # Compute im_conv = f*I
  return im_conv
```

Median filter: Implement median filtering. Your function should take an image *I* and a filter size *n* as input and return an image denoised with an *n*x*n* medial filter. You may sort with built in functions. Make sure the filter has an odd size, i.e. if the user gives an even *n* just increase the size by 1. You can handle the boundary of *I* in any reasonable way of your choice.

```
def median_filtering(I, n):
  # Handle boundary of I, e.g. pad I according to size n
  # Denoise image with an nxn median filter
  return denoised_im
```

A1b) Image Denoising: Compare Box filter, Gaussian Filter, and Median Filter

Compare the results of box filter, Gaussian filter, and median filter. Use the included grayscale image '*lena.png*' with varying amount of added noise. You can add noise to the image with the following python code (https://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.random.randn.html):

```
# Add Gaussian noise with mean 0 and std deviation sigma to image 'im'
sigma = 0.1
im_noise = im + sigma*np.random.randn(*im.shape)
```

Add several levels of noise (e.g. low, medium, high) to the original image and compare the results of box, Gaussian, and median filters. How do you choose the size of box filter, σ of the Gaussian, and size of the median filter to handle an increasing amount of noise? Make sure to use your implementation of convolution for box and Gaussian filtering (and your implementation of median filtering). How can you compare your results, given you have a clean (ground-truth) image '*lena.png*' that is uncorrupted by noise? Include results, explanations, and discussion in your report. What can you say about the strengths/weakness of these denoising methods?

A2) Derivative Filters

Compute derivative images (with respect to x and with respect to y) of the uploaded image '*cameraman.png*' using the separable derivative filter of your choice, e.g. [-1 0 1] and $[-1\ 0\ 1]^T$. You can hardcode the derivative filter, but use your implementation of convolution. Compute the gradient magnitude image. Create binary edge images from the gradient magnitude image using several thresholds. Explain how the derivative filter is a discrete approximation of a derivative. Show all results and discuss the outcome of various thresholds.

A3) Template Matching

The uploaded image '*multiplekeys.png*' is a single image containing multiple instances of keys.

Perform the following steps:

- Threshold the original key image so that the background is [0.0] and keys appear as [1.0]. You should have a binary image with only [0.0] (background) and [1.0] (keys).
- Choose your favorite key and crop with a narrow boundary. Use this image as your template.
- Modify your template by setting background pixels to [-1.0], so that you have [+1.0] for the key and [-1.0] for background. The reason for creating a signed template is improved matching performance.
- Implement cross-correlation with the binary input image and your new signed template image.

- Please recall that this results in a peak (maximum) if the template matches the specific image region which you selected for your template.
- Do a pass through the correlation image to detect the maximum peak value and its (x,y) location. You may mark this location with overlay of a circle or just manually painting some arrow or similar. Discuss if this location matches your expectation, and also discuss what happens to the peak of the other keys.

In your report, show the original image, peak image, and your template which was used for correlation.