Introduction to Computer Vision (ECSE 415)
Assignment 1: Image Filtering

Due date: 11:59PM, February 1st, 2021

Please submit your assignment solutions electronically via the myCourses assignment dropbox. The submission should include a single jupyter notebook. More details on the format of the submission can be found below. Submissions that do not follow the format will be penalized 10%. Attempt all parts of this assignment. The assignment will be graded out of total of **46 points**. You can use OpenCV and Numpy library functions for all parts of the assignment except stated otherwise. Students are expected to write their own code. (Academic integrity guidelines can be found at https://www.mcgill.ca/students/srr/academicrights/integrity). Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Submission Instructions

1. Submit a single jupyter notebook consisting of the solution of the entire assignment.
2. Comment your code appropriately.
3. Do not forget to run Markdown cells.
4. Do not submit input/output images. Output images should be displayed in the jupyter notebook itself. Assume input images are kept in the same directory as the codes.
5. Make sure that the submitted code is running without error. Add a README file if required.
6. If external libraries were used in your code please specify their name and version in the README file.
7. Answers to reasoning questions should be comprehensive but concise.
8. Submissions that do not follow the format will be penalized 10%.

Note: For this assignment, we will work with grayscale images. DO NOT forget to convert your images to GrayScale from RGB images.

1 Thresholding (12 Points)

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images. Here, each pixel in an image is replaced with a foreground label (i.e. a white pixel with 255 value) if the image intensity $I_{i,j}$ satisfies some pre-defined condition (Ex. if $I_{i,j} > T$) in relation to threshold values (Ex. T, T1, T2), or with a background label (i.e. a black pixel with 0 value) otherwise.

Simple Binary Thresholding:

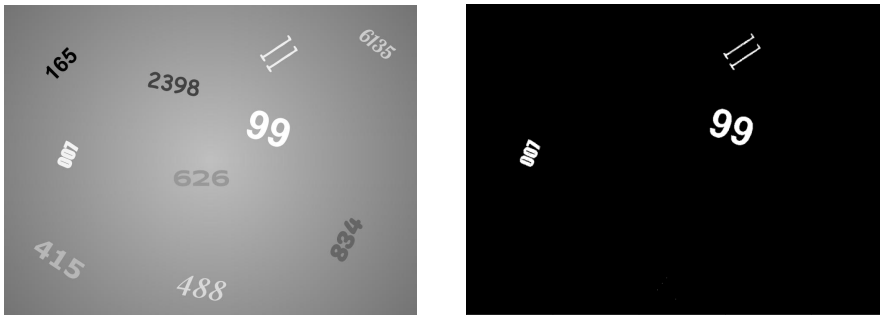
$$S_{i,j} = \begin{cases} 255 & \text{if } I_{i,j} > T; \quad (1) \\ 0 & \text{otherwise.} \quad (2) \end{cases}$$

Window Binary Thresholding:

$$S_{i,j} = \begin{cases} 255 & \text{if } T1 < I_{i,j} < T2; \quad (3) \\ 0 & \text{otherwise.} \quad (4) \end{cases}$$

You are given an image named "numbers.jpg" (Figure 1(a)) which contains multiple different multi-digit numbers. Your task is to threshold the image using the pre-defined conditions for thresholding defined above.

Note that you are **not** allowed to use the openCV **cv2.threshold** function for this task. Instead implement thresholding using basic python or numpy functions.



(a) (b)

Figure 1: (a) Input image for thresholding, (b) Example of output image of thresholding. Note that only numbers "007", "11", and "99" are segmented (foreground pixels).

1. Threshold the image at three different thresholds 1) 45 2) 90 and 3) 160 using simple binary thresholding as defined above. Display thresholded images at these thresholds. **(3 points)**

2. Write your observations about thresholded images at different thresholds. How many and which numbers are segmented at each threshold? Note: A number is considered as segmented if all digits of that number are clearly visible as foreground or background in the thresholded image. What else do you observe at each threshold? **(3 points)**

3. Threshold the image using Window binary thresholding using three different ranges of thresholds. 1) T1=55 and T2=110, 2) T1=110 and T2=160, 3) T1=55 and T2=160. Write your observations. Display images at these different thresholds. How many and which numbers are segmented at each threshold? **(3 points)**

4. In a practical application, we vary the value of the hyper-parameters (here, the threshold values) for any of the above-mentioned thresholding methods, such that we get the desired output. Find a threshold value such that only numbers "007", "11", and "99" are segmented (i.e. considered as foreground - white pixel - 255 value). See Figure 1(b). Report your finding and display corresponding thresholded images for at least three different threshold values, and write how it helped you in narrowing down the desired hyper-parameter value. Note that the emphasis in to not get the exact output but to explore different hyper-parameters and report your finding. **(3 points)**

2 Denoising (18 Points)

You are **not allowed** to use OpenCV/Scikit-image functions for this section **unless specified otherwise**. For this question, you are expected to **write your own code in NumPy for convolution. (4 points)**

Refer to lecture slides on Image filtering covered during class. For handling boundary conditions, you are expected to use zero padding (covered during lecture/tutorial).

You are given a clean image named 'Tower' (Figure 2(a)) and an image corrupted by additive white Gaussian noise (Figure 2(b)).

Apply the following filtering operations:

1. Filter the noisy image using a 3×3 Gaussian filter with variance equal to 2. Display filtered image along with original image. **(2 points)**

2. Filter the noisy image using a box filter of the same size. Display filtered image along with original image. **(2 points)**

(a) (b) (c)

Figure 2: Input images for denoising. (a) clean image [1] (b) image corrupted with Gaussian noise (c) image corrupted with salt and pepper noise.

3. Compare the Peak-Signal-to-Noise-Ratio (PSNR) of both of the denoised images to that of the clean image and state which method gives the superior result. Use the PSNR function provided by opencv/scikit-image. **(3 points)**

You are also given an image corrupted by salt and pepper noise (Figure 2(c)). Apply the following filtering operations:

4. Filter the noisy image using the same Gaussian filter as used in the previous question. Display filtered image along with original image. **(2 points)**

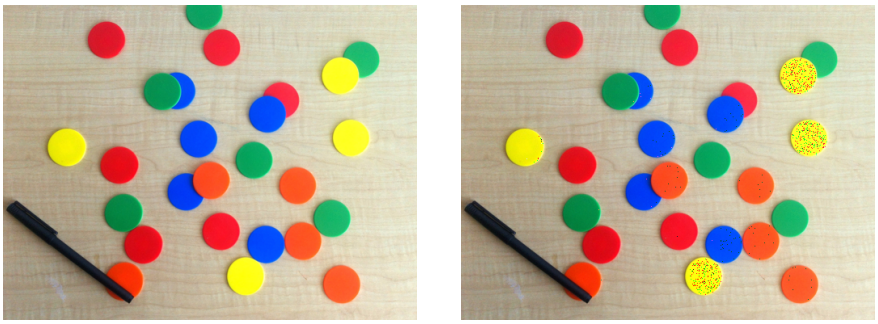
5. Filter the noisy image using a median filter of the same size. Display filtered image along with original image. **(2 points)**

6. Compare the PSNR of both of the denoised images to that of the clean image and state which method gives a better result. Use the PSNR function provided by opencv/scikit-image. **(3 points)**

3 Sobel edge detector (16 Points)

In this question, you will assess the effect of varying the kernel size on the results of an edge detection algorithm. You will detect edges in a clean image named, 'circles' (Figure 3(a)). You are **allowed** to use OpenCV/Scikit-image functions for this section, **including the convolution function**.

- Apply a Sobel edge detector with the kernel size of 3×3, 5×5 and 7×7 to the image. Threshold the filtered image to detect edges. Use two values of thresholds: 10% and 20% of the maximum pixel value (magnitude) of



(a) (b)

Figure 3: "Circles" [2]: Input image for edge detection. (a) clean image (b) image corrupted with Gaussian noise.

the filtered image. Display phase, magnitude, and thresholded images for different kernel sizes and different thresholds. **(6 points)**

- Comment on the effect of filter size on the output. **(2 points)**

Next, you will evaluate the effect of denoising prior to edge detection. For the following questions, you will use noisy image as shown in Figure 3(b).

- Apply a Sobel edge detector with the kernel size of 3×3. Threshold the filtered image to detect edges. Use two values of thresholds: 10% and 20% of the maximum pixel value in magnitude of the filtered image. Display phase, magnitude, and thresholded images for different thresholds. **(3 points)**
- Denoise the image with a 3×3 box filter and then apply the same Sobel edge detector, with the same values of the thresholds, from the previous question. Display original and filtered image side by side. Display phase, magnitude, and thresholded images for different thresholds. **(3 points)**
- Comment on the effectiveness of using denoising prior to edge detection. **(2 points)**