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

Due date: 11:59PM, October 7, 2019

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 in the last section. 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 **70 points**.

Students are expected to write their own code. (Academic integrity guidelines can be found at `https://www.mcgill.ca/students/srr/academicrights/integrity`).

Note1: Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Note2: Please make sure that you convert given images into grayscale images before doing any processing.

1 Thresholding

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images. We will consider a simple case, binary thresholding, which we studied in Tutorial-3. Here, each pixel in an image is replaced with a foreground label (i.e. a white pixel with 255 value) if the image intensity $I_{i,j}$ is greater than some fixed constant T (that is, $I_{i,j} > T$), or with a background label (i.e. a black pixel with 0 value) if the image intensity is greater than that constant.

You are given an image named "numbers.jpg" (Figure 1(a)) which contains multiple different multi-digit numbers. Your task is to threshold image at different thresholds using binary thresholding define above. (Use cv2.threshold function for this task)

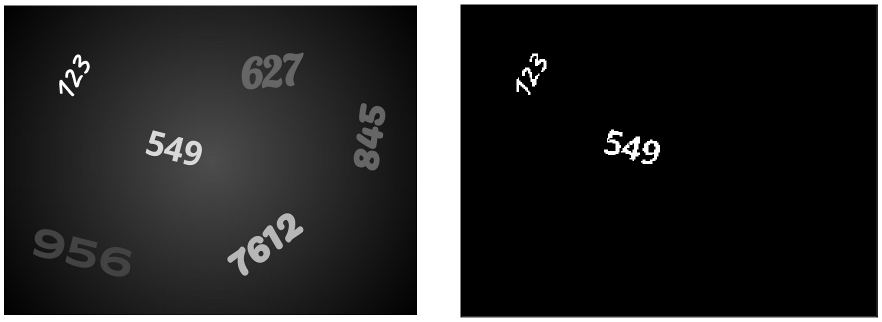


Figure 1: (a) Input image for thresholding, (b) Example of output image of thresholding. Note that only numbers "123" and "549" are segmented (foreground pixels).

1. Threshold image at three different thresholds 1) 55 2) 90 and 3) 150 **(3 points)**
2. Write your observations about thresholded images at different thresholds. How many and which numbers are segmented at each threshold? (A number is considered as segmented if all digits of that number are considered as foreground in the thresholded image) What else do you observe at each threshold? **(3 points)**
3. In a practical application, we vary the value of hyper-parameter (here, the threshold value) such that we get the desired output. Find a threshold such that only numbers "123" and "549" are segmented (i.e. considered as foreground - white pixel - 255 value). See Figure 1(b). **(3 points)**

2 Filtering

2.1 Denoising

You are given a clean image named, 'Circles.png' (Figure 2(a)) and an image corrupted by additive white Gaussian noise ('Circles_gauss.png') (Figure 2(b)). Apply following filtering operations:

1. Filter the noisy image using a 5×5 Gaussian filter. **(2 points)** (Use cv2.GaussianBlur - Link)
2. Filter the noisy image using a box filter of the same size. **(2 points)** (Use cv2.blur - Link)
3. Compare the PSNR (Eq.1) of both of the denoised images to that of the clean image and state which method gives the superior result. **(2 points)**

(you can use cv2.PSNR - Link)

$$PSNR(\text{I1}, \text{I2}) = 10 * \log_{10}(\frac{255^2}{MSE(\text{I1}, \text{I2})}) \tag{1}$$

$$MSE(\text{I1}, \text{I2}) = mean((I1 - I2)^2) \tag{2}$$

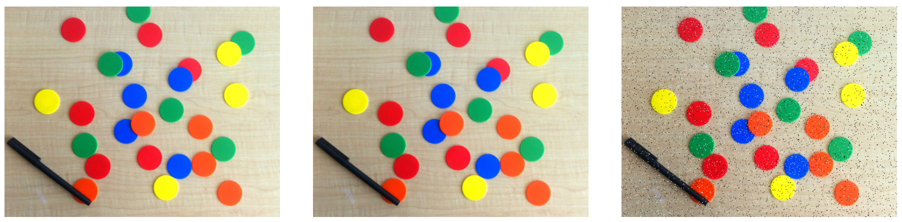
4. Now, increase the size of filter to 1) 9×9 and 13×13 and repeat above mentioned filtering (Gaussian and Box). What do you observe? Does PSNR increase for both filtering? Do you think visually image quality increases? **(3 points)**

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

5. Filter the noisy image using a 5×5 Gaussian filter. **(2 points)** (Use cv2.GaussianBlur - Link)
6. Filter the noisy image using a median filter of the same size. **(2 points)** (Use cv2.medianBlur Link)
7. Compare the PSNR of both of the denoised images to that of the clean image and state which method gives a better result. **(2 points)** (you can use cv2.PSNR - Link)
8. Now, increase the size of filter to 1) 9×9 and 13×13 and repeat above mentioned filtering (Gaussian and median). What do you observe? Does PSNR increase for both filtering? Do you think visually image quality increases? **(3 points)**

2.2 Sharpening

Sharpening of an image (I) is used to enhance line structures or other details in an image. One of the simplest way of recovering details (D) of an image is by



(a) (b) (c)

Figure 2: Input images for denosing. (a) clean image (b) image corrupted with Gaussian noise (c) image corrupted with salt and pepper noise. (Don't forget to convert these images into grayscale images)

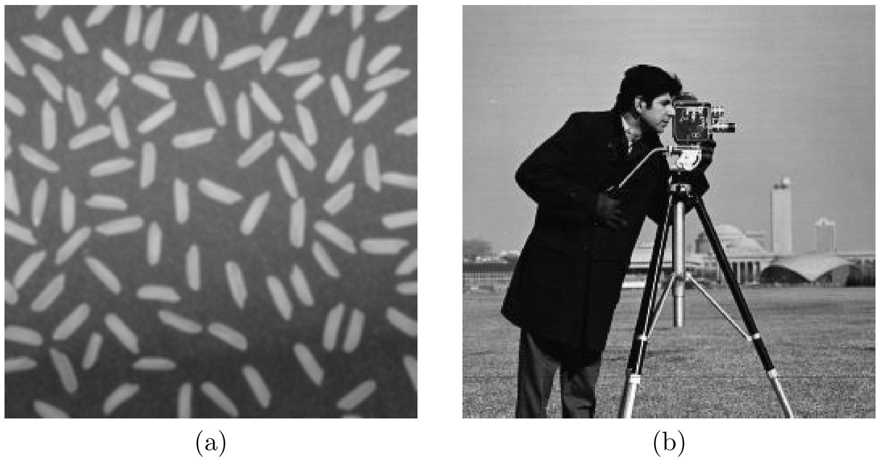


Figure 3: Input images for sharpening and edge detection. (a) rice (b) cameraman.

calculating difference between an image (I) and its blurred version (B). We then add this details (D) in the original image (I) and generate sharpened image (S).

In short, Sharpening is defined as follows:

$$\text{details (D)} = \text{original image (I)} - \text{blurred image (B)}$$
$$\text{sharpened image (S)} = \text{original image (I)} + \text{details (D)}$$

1. Sharpen the given image 'rice.png' (see Figure 3(a)) where,
    - the blurred image is generated using 7×7 box filter. **(3 points)** (Use cv2.blur - Link)
    - the blurred image is generated using 7×7 Gaussian filter. **(3 points)** (Use cv2.GaussianBlur - Link)
2. Which of the two methods is expected to give better results and why? Can you observe expected result? **(3 points)**

3 Edge detection

3.1 Sobel edge detector

1. Effect of sharpening on edge detection
    - Apply a Sobel edge detector to the image 'rice.png' (Figure 3(a)). Use kernel size of 5. **(2 points)** (Use cv2.Sobel - Link)

- Apply a Sobel edge detector to the two previously-sharpened images (using Gaussian and Box filters in Section-2.2) with same kernel size. **(2 points)**
- Remember that, we can extract the magnitude and phase of edges after applying sobel operator (refer Tutorial-3). First, recover the magnitude and then apply Thresholding (Section-1) on the magnitude image. Apply two different thresholds: 200 and 800. As you will apply thresholding on the three generated images from the last two questions, this should result in a total of 6 thresholded images. **(4 points)**
- Comment on the effectiveness of using sharpening prior to the edge detection. **(2 points)**
- Comment on the effect of the different threshold values. **(2 points)**

2. Effect of Denoising on Edge Detection:

    - Apply a Sobel edge detector (Filter size 5) to the image 'cameramen.jpg' (see Figure 3(b)). Threshold the magnitude using a threshold value of 1500. **(1 points)**
    - First denoise image with a 5×5 gaussian filter and then apply a Sobel edge detector. Use the same value of threshold (1500). **(2 points)**
    - Comment on the effectiveness of using denoising prior to edge detection. **(2 points)**

3.2 Laplacian of Gaussian

1. Apply a 5×5 Laplacian of Gaussian *edge detector* to the image 'cameraman.jpg' (see Figure 3(b)). **(2 points)** (Use cv2.Laplacian - Link)
2. Now increase the size of the filter to 15×15. Write your observations about the edge maps. **(3 points)**

3.3 Canny Edge Detection

For this section, experiments will be performed on the 'dolphin.jpg' (Figure 4(a)) image.

1. Briefly describe 4 main steps of canny edge detection. **(2 points)**
2. As saw in Tutorial-3, 3 main hyperparameters of Canny Edge detection are the Gaussian Smoothing Kernel size (K), and the Lower (L) and Higher (H) Thresholds used for Hysteresis. In this section, we will observe the effect of changing these hyperparameters. You will experiment on 3 different values for all 3 parameters (K = {5,9,13}, L = {10,30,50}, H = {100, 150,

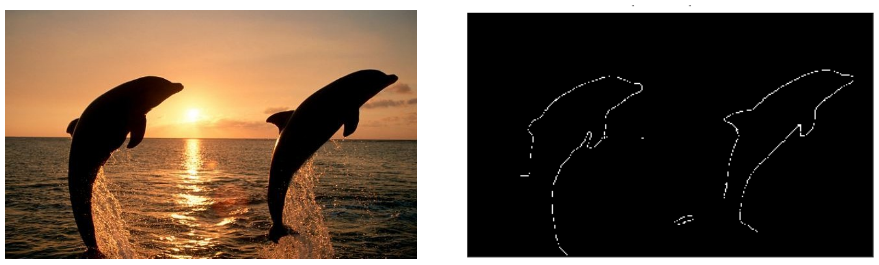

(a) (b)

Figure 4: (a) Input image for Canny (don't forget to convert this into grayscale image) (b) Example of canny edge detection output. Note that only dolphin edges are detected.

200}). Vary the values of each hyper-parameter and keep other hyper-parameters constant. Do this procedure for all combination of hyper-parameters mentioned above. This should results in total 27 triplets of hyper-parameters. E.g. (K,L,U) = {(5,10,100), (5,10,150), (9,10,200), ... }. Use canny edge detection (cv2.GaussianBlur - Link and cv2.Canny - Link) for each of these triplets. **(4 points)**

3. Comment on how changing values of each hyper-parameters (K,L,U) effects the overall edge detection. Is there is any relationship between any hyper-parameters? **(3 points)**

4. Find a value of each hyper-parameter such that only dolphin edges are detected. (Figure 4(b)) **(3 points)**

4 Instruction for the Report

Please recall submissions that do not follow the format will be penalized 10%.

4.1 Report

1. Submit one single jupyter notebook for the whole assignment. It should content everything (code, output, and answers to reasoning questions)

2. Write answers section-wise.

3. Numbering of answers should match exactly as the questions.

4. Comment your code appropriately.

5. Answers to reasoning questions should be brief but comprehensive. Unnecessarily lengthy answers will be penalized. Use *markdown cell* type in jupyter notebook to write the answers.

6. If external libraries were used in your code please specify its name and version in the README file.

7. Run the whole jupyter notebook one last time using *Cell- ¿ Run All*. This will run all cells in the jupyter notebook and show all outputs. We expect that when we run your jupyter notebook, we should be able to reproduce your results.