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

Due date: 11:59PM, February 11, 2020

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 **66 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.

1 Submission Instructions

1. Submit a single jupyter notebook consisting 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 a 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 its 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%.

2 Denoising

You are given a clean image named, 'lighthouse' (Figure 1(a)) and an image corrupted by additive white Gaussian noise (Figure 1(b)). Apply following filtering operations:

(a) (b) (c)

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

1. Filter the noisy image using a 5×5 Gaussian filter with variance equal to 2. **(3 points)**
2. Filter the noisy image using a box filter of the same size. **(3 points)**
3. Compare the PSNR of both of the denoised images to that of the clean image and state which method gives the superior result. (Use PSNR function provided by opencv) **(3 points)**

You are also given an image corrupted by salt and pepper noise (Figure 1(c)). Apply the following filtering operations:

4. Filter the noisy image using the same Gaussian filter as used in the previous question. **(3 points)**
5. Filter the noisy image using a median filter of the same size. **(3 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. **(3 points)**

3 Sobel edge detector

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, 'cameraman' (Figure 2(a)).

- Apply a Sobel edge detector with the filter 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 in the filtered image. **(4 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 2(b).

- Apply a Sobel edge detector with the filter 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 the filtered image. **(4 points)**
- Denoise the image with a 3×3 box filter and then apply a Sobel edge detector. Use the same values of the thresholds from the previous question. **(4 points)**
- Comment on the effectiveness of using denoising prior to edge detection. **(3 points)**

4 Laplacian of Gaussian

For this question, you will also use the image 'cameraman' (Figure 2(a))

1. Filter the image using a 5×5 Laplacian of Gaussian kernel. **(3 points)**
2. Write your own code to detect edges in the above result. Recall that the zero-crossings in the filtered image indicate edges. **(4 points)**

(a) (b)

Figure 2: Input image for edge detection. (a) clean image (b) image corrupted with Gaussian noise.

5 Derivative of Gaussian

1. Create a Derivative of Gaussian kernel as follows. First create a Gaussian kernel of size 7×7 using the code given below.

```
import cv2
import numpy as np
temp = cv2.getGaussianKernel(7, -1)
Gauss = np.outer(temp, temp.transpose())
```

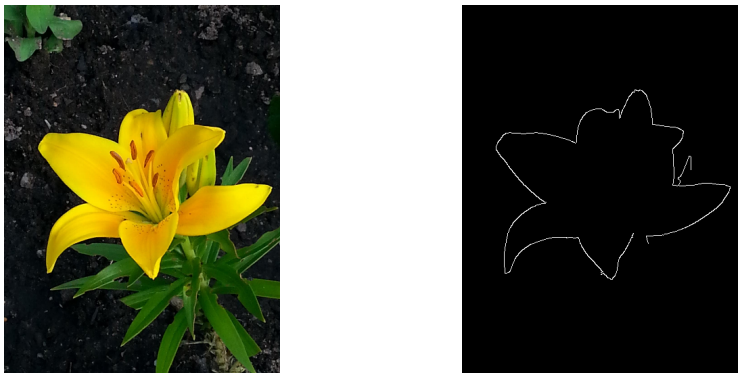
   Then apply a 3×3 Sobel kernel in x and y direction separately to the Gaussian kernel. The resultant kernels are Derivative of Gaussian kernels. **(3 points)**

   Apply a Derivative of Gaussian kernel created above to the noisy image shown in Figure 2(b). Threshold the resultant image with two values of thresholds: 10% and 20% of the maximum pixel value in the filtered image. **(3 points)**

2. Denoise the same image with 5×5 Gaussian kernel. Apply 3×3 Sobel edge detector on the denoised image. Threshold the resultant image with two values of thresholds: 10% and 20% of the maximum pixel value in the filtered image. **(4 points)**

3. Apply 3×3 Sobel edge detector on the original noisy image. Threshold the resultant image with two values of thresholds: 10% and 20% of the maximum pixel value in the filtered image. **(3 points)**

4. State your observations about the differences/similarities in the above three results. **(3 points)**



(a) (b)

Figure 3: (a) Input image for canny edge detection (b) expected output.

6 Canny Edge Detection

1. Use canny edge detector to detect edges in 'yellowlily' image shown in Figure 3(a). Tune kernel size, lower and upper thresholds such that the final output is similar to the one shown in 3(b). **(5 points)**

2. Comment on how changing values of kernel size, lower and upper thresholds affect the overall edge detection. **(3 points)**