Digital Image Processing (CSE/ECE 478)
Monsoon-2020
Assignment-2
Posted on: 09/09/2020
Due on: 07/10/2020

Introduction

1. All images in this assignment should be considered to be used as grayscale unless specified otherwise. Use the appropriate function to convert colour images to grayscale, or read the images directly as grayscale.
2. Follow the specified repository structure as in previous assignments.
3. `src` should contain the Jupyter notebooks used for the assignment.
4. `images` should contain images used for the questions.
5. Make sure you run your Jupyter notebook before committing, to save all outputs.
6. Make sure you commit and push your work regularly.

---

1. (50 points) The implementation of linear spatial filters requires moving a mask centered at each pixel of the image, computing sum of products of mask coefficients and corresponding pixels.
    1. Implement an algorithm for low-pass filtering a grayscale image by moving a $k \times k$ averaging filter of the form `ones((k,k))/(k^2)`.
    2. As the filter is moved from one spatial location to the next one, the filter window shares many common pixels in adjacent neighborhoods. Exploit this observation and implement a more efficient version of averaging filter.
    3. To appreciate the benefits of doing so, generate a plot of k vs run-time for various sized images. The plot diagram should contain a line plot for each image size you pick. Use different marker types to distinguish the default implementation and improved implementation. Just to give you a rough idea, look at `https://imgur.com/a/0HtYlTE`.
    4. Utilize the observation similar to above to implement an efficient version of a $k \times k$ median filter. Generate a plot figure similar to previous question.
    5. Using the denoising concepts discussed in the class, try to remove noise from **Noisy.jpg**

---

2. (60 points) Edge Detection

    1. Read about Canny Edge detector. Apply the Canny edge detector (use inbuilt *cv2.Canny* function for this task) to `bell.jpg`.
    2. Tweak the values of the arguments (*minVal* and *maxVal*), and report the values that give the best results for each image. (Hint: Try to detect as many edges in the bell as possible, while avoiding edges in the background.).
    3. Consider **Roberts, Prewitt, Sobel and Laplacian** filters discussed in the class. Implement and apply these filters on `kobe.png` and make observations upon comparing their outputs. Compare these with the output of Canny edge detector on the same image.

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Figure 1: Prewitt, Roberts and Sobel filters

$$\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{-4} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{-8} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Figure 2: Two forms of the laplacian filter

    4. What will the 5×5 variants of **Sobel and Prewitt** filters look like? Apply these larger filters on `kobe.png` and make observations upon comparing their outputs with the corresponding smaller filters.
    5. Add noise to the input image above using Gaussian sampling. Study the effect of applying the above filters(from 2^{nd} part) on noise-affected inputs.

3. (40 points) Bilateral Filtering

    1. Implement Bilateral Filtering and apply it on **mountain.jpg**
    2. In low light imaging, images tend to be noisy and lose sharp edges. However, with flash, there is an unpleasant direct-lighting effect. Here, a **cross bilateral filter** can be used with the range and spatial filters acting on two different images.
    3. By combining a flash photograph **cake-flash.jpg** and a no-flash photograph **cake-noflash.jpg**, render a new photograph **cake-out.jpg** that has both the warm lighting of the no-flash picture and the crisp details of the flash image.

Figure 3: flash, no-flash and output images

Note: This question has to be done on RGB images. For applying filters on RGB images you can do the same filtering operation used on grayscale image for each channel (i.e., R,G,B) separately, one by one.