CSCI 3290 Computational Imaging and Vision

Spring 2020, Assignment 1 – Image Sharpening

Due Date: Feb. 6, 2020 (23:59:59)

Submission via blackboard.cuhk.edu.hk

Late submission penalty: 10 marks deduction per day.

PLAGIARISM Penalty: Whole Course Failed

I. Background

This assignment aims at practicing the foundation of digital image processing and computer vision, image filtering. Filtering is a technique for modifying or enhancing an image. We can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and denoising. We will image sharpening in this assignment.

II. Algorithms

Filtering is a neighborhood operation, in which the value of the output image is determined by applying some algorithm to the values of the pixels inside the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. Linear filtering is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood. We will use linear filtering technique to do the image sharpening.

• Convolution

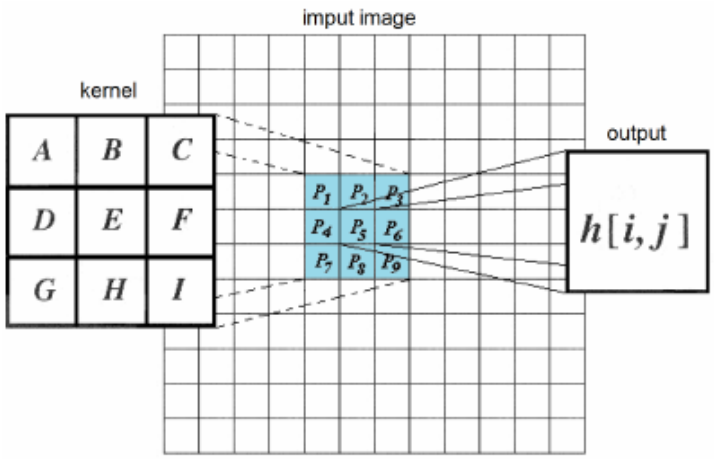

Figure 1. Convolution process for a 3 × 3 convolution kernel.

Linear filtering of an image is accomplished through an operation called convolution. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels.

The matrix of weights is called the convolution kernel, also known as the filter. As shown in Figure 1, the computation of the output image pixel value h[i, j] at this location is done as:

$$h[i, j] = A \times P_1 + B \times P_2 + C \times P_3 + D \times P_4 + E \times P_5 + F \times P_6 + G \times P_7 + H \times P_8 + I \times P_9$$

To get the whole output image, we should slide the kernel over the whole input image step by step. For example, Figure 2 shows the kernel sliding process of applying a 3x3 kernel (shadow region) on a 4x4 input image (bottom blue region) to get the output image (top green region).

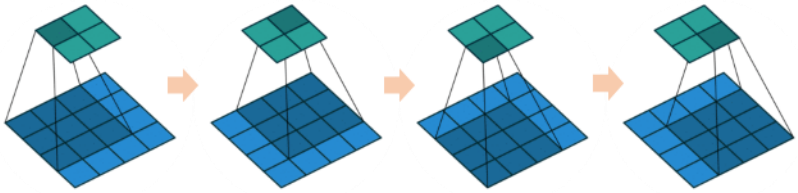

Figure 2. Kernel sliding process.

• Kernel generation

Applying different kernels in convolution can achieve different editing effects. In this assignment we will use the Gaussian kernel, specifically, 2D Gaussian kernel. Mathematically, the 2D Gaussian kernel is computed as:

$$G_{2D}(x, y; \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{y^2}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

where x and y are the horizontal and vertical distance to the center of the kernel, respectively; σ is the standard deviation (*always positive*) that controls the width of the Gaussian kernel. The above equation is used to compute the infinite size gaussian kernels, however, practically we always use finite size kernels, like 3*3 or 5*5. Therefore, we need to normalize the kernel to make the sum of all the kernel elements values to be one. We can do it like this:

$$K(x, y; \sigma) = \frac{1}{\sum_{i=0,j=0}^{i=m,j=n} G\_2D\ (i, j; \sigma)} * G_{2D}(x, y; \sigma),$$

where $m * n$ is the kernel size. In this assignment we only account for the square shape kernel, so $m == n$ all the time, and we only account for the kernel size to be odd number, like 1, 3, 5, 7, …

• Overall pipeline

Having the above definition, we can do the image sharpening by first convolve the input image using a Gaussian kernel to get the smoothed image, then use the input image to minus the smoothed image to get the detail map, and finally add the detail map to the original input image to get the sharpened image. The pseudo-code is as following:

1. Read the input image by *imread()* function (provided).
2. Generate the Gaussian kernel by *gaussian_kernel()* function (need to be implemented) with the given kernel size and standard deviation σ.
3. Convolve the input image by *conv()* function (need to be implemented) with the generated kernel to get the smoothed image.
4. Generate the sharpened image by *sharpen()* function (need to be implemented) with the generated smoothed image and original input image like this:
   i. crop input_image to get input_image_crop that has the same size with smoothed_image.
   ii. detail_map = input_image_crop – smoothed_image
   iii. sharpened_image = input_image_crop + detail_map
   iv. *return* sharpened_image
5. Write the sharpened image by *imwrite()* function (provided).

This all the procedure of our assignment. There are parameters, standard deviation σ and kernel shape $m * m$, to control the degree of sharpen effect in the image sharpening process.

III. Assignment Details

In this assignment, you are required to implement the algorithm in **Python 3.4+.** We will provide a skeleton code "*sharpening.py*" that leaves the three major functions empty for you to implement.

1. The program should support variable kernel size and standard deviation σ. The kernel shape is assumed to be square and odd number value integers, like 1, 3, 5, 7, …, and the standard deviation σ is assumed to be positive float point number.

2. The assignment assumes the input image is grayscale in one channel, so the *imread()* function will return a H*W 2D array. And to avoid complexity, we only account for and support PNG format grayscale images. We will provide some example images for you to test your program. You need to install *imageio* to run the skeleton code by following instruction:

```
> pip install imageio
> pip install numpy
Or
> pip3 install imageio
> pip3 install numpy
```

3. You should implement the functions in ***pure Python 3***. Except for the skeleton code provided libraries, do ***not*** import any other third-party libraries (like *SciPy* and *OpenCV*)!

4. Do not modify the provided function, except for the three functions that need to be implemented.

5. You should make the program runnable without errors.

6. Your source code should be named "*studentID_ sharpening.py*".

7. The command line to run your program should have the same format as the skeleton code:

> `python3` studentID_ sharpening.py --input /PATH/TO/INPUT/IMAGE --kernel KERNEL_SIZE --sigma SIGMA --output /PATH/TO/OUTPUT/IMAGE

IV. Marks

Functions	**Marks**
gaussian_kernel()	40
conv()	50
sharpen()	10

If you do not follow the Sec.III details, marks will be deducted.

V. Submission Guidelines

1. In all your source files, type your full name and student ID, just like:

```
#
# CSCI3290 Computational Imaging and Vision *
# --- Declaration --- *
# I declare that the assignment here submitted is original except for source
# material explicitly acknowledged. I also acknowledge that I am aware of
# University policy and regulations on honesty in academic work, and of the
# disciplinary guidelines and procedures applicable to breaches of such policy
# and regulations, as contained in the website
# http://www.cuhk.edu.hk/policy/academichonesty/ *
# Assignment 1
# Name :
# Student ID :
# Email Addr :
#
```

Missing of these pieces of essential information will lead to mark deduction.

2. You are **required** to write your programs using ***pure Python 3.4+*** language without importing any other third-party libraries, since this allows your code to be compatible on different platforms.
3. You are required to send your homework to the blackboard system. https://blackboard.cuhk.edu.hk/
4. 10 marks will be deduced per day delayed *including public holidays & Sundays.* You will fail the course if you copy others' work.