ABHISHEK BASU
658369163

CS 415 Mini Project 1

1 Question Answering

Q1. (**1) What is the goal of computer vision?**

Ans:

The primary aim of computer vision is to empower machines to interpret and understand visual data from the environment, akin to human visual perception. This field analyzes images and videos to derive meaningful information, enabling machines to make decisions and perform tasks based on visual input.

(2) Please list three computer vision tasks (for example, face detection) and their respective applications.

Ans:

1. **Face Detection**
   - **Application**: Employed in security systems for surveillance, in social media for automatic tagging of users, and in mobile devices for unlocking screens through facial recognition.
2. **Object Recognition**
   - **Application**: Used in self-driving cars to identify and track pedestrians, traffic signals, and other vehicles, as well as in retail environments for managing inventory and improving customer experience.
3. **Image Segmentation**
   - **Application**: Applied in the medical field for analyzing images to locate tumors or other abnormalities and in robotics for helping machines understand and navigate their surroundings.

(3) What is a (digital) RGB image?(10 points)

Ans:

A digital RGB image is a format that represents colors using the RGB color model, where colors are generated by mixing red, green, and blue light at different intensities. Each pixel in an RGB image is characterized by three values indicating the levels of red, green, and blue, typically ranging from 0 to 255. This combination allows for the representation of millions of colors, making it suitable for detailed image rendering.

Q2. (1) Please briefly describe the process of linear filtering.

Ans:

Linear filtering involves modifying or enhancing a signal (such as audio or images) using a linear operator defined by a filter kernel or mask. The steps include:

1. **Filter Kernel Definition**: Choose a small matrix that contains weights for combining nearby values. Examples of kernels include Gaussian, Sobel, and average filters.
2. **Sliding the Kernel**: The filter is moved across the input signal or image.
3. **Weighted Sum Calculation**: For each position of the kernel, multiply the overlapping values from the input with the corresponding weights in the kernel.
4. **Result Aggregation**: Sum the weighted values to produce a new value for the output

at that position.

5. **Output Creation**: Repeat until the entire input is processed, yielding the final filtered output.

(2) What are the commonalities and differences between (cross) correlation and convolution? (15 points)
Ans:

Commonality:

1. **Linear Operations**: Both correlation and convolution are linear operations applied to signals.
2. **Kernel Application**: Each method involves sliding a kernel over an input signal or image and computing a weighted sum of overlapping values.
3. **Applications**: Both techniques are widely used in signal processing, image processing, and machine learning for tasks like filtering, feature extraction, and pattern recognition.

Differences Between Cross-Correlation and Convolution

1. **Kernel Orientation**:
   - **Cross-Correlation**: The kernel is applied in its original orientation as it slides over the input signal.
   - **Convolution**: The kernel is flipped (180 degrees) before being applied to the input signal.
2. **Interpretation of Output**:
   - **Cross-Correlation**: This operation measures how similar two signals are as they shift over the other, indicating their correlation.
   - **Convolution**: This operation combines two signals to create a new signal, often used for tasks like filtering and smoothing.
3. **Applications**:
   - **Cross-Correlation**: Typically used in fields like pattern recognition, image registration, and time series analysis.
   - **Convolution**: Commonly applied in image processing for tasks like blurring, sharpening, and edge detection.
4. **Result Characteristics**:
   - The outputs from cross-correlation and convolution can behave differently, especially in how they affect frequency content. Convolution can transform the input signal, while cross-correlation focuses on the alignment between signals.

ABHISHEK BASU
658369163

Q3. Below are a 3x3 grayscale input image (left) and a 3x3 kernel (right). Please manually perform correlation and convolution. Zero padding should be used to make the size of the output image the same as that of the input image. (15 points)

1 0 2
2 2 1
2 1 0

2 1 1
1 2 0
0 0 1

Image:
1 0 2
2 2 1
2 1 0

Kernel:
2 1 1
1 2 0
0 0 1

Zero-Padded Image:
0 0 0 0 0
0 1 0 2 0
0 2 2 1 0
0 2 1 0 0
0 0 0 0 0

Correlation:
Position (0,0): (0*2 + 0*1 + 0*1 + 0*1 + 1*2 + 0*0 + 0*0 + 2*0 + 2*1) = 4
Position (0,1): (0*2 + 0*1 + 0*1 + 1*1 + 0*2 + 2*0 + 2*0 + 2*0 + 1*1) = 2
Position (0,2): (0*2 + 0*2 + 0*1 + 0*1 + 2*2 + 0*0 + 2*0 + 1*0 + 0*1) = 4
Position (1,0): (0*2 + 1*1 + 0*1 + 0*1 + 2*2 + 2*0 + 0*0 + 2*0 + 1*1) = 6
Position (1,1): (1*2 + 0*1 + 2*1 + 2*1 + 2*2 + 1*0 + 2*0 + 1*0 + 0*1) = 10
Position (1,2): (0*2 + 2*1 + 0*1 + 2*1 + 1*2 + 0*0 + 1*0 + 0*0 + 0*1) = 6
Position (2,0): (0*2 + 2*1 + 2*1 + 0*1 + 2*2 + 1*0 + 0*0 + 0*0 + 0*1) = 8
Position (2,1): (2*2 + 2*1 + 1*1 + 2*1 + 1*2 + 0*0 + 0*0 + 0*0 + 0*1) = 11

ABHISHEK BASU
658369163

Position (2,2): (2*2 + 1*1 + 0*1 + 1*1 + 0*2 + 0*0 + 0*0 + 0*0 + 0*1) = 6

Correlation Result
[[4 2 4]
[6 10 6]
[8 11 6]]

Convolution
In convolution, we flip the kernel.
1 0 0
0 2 1
1 1 2

Calculating Convolution:
Position (0,0): (0*1 + 0*0 + 0*0 + 0*0 + 1*2 + 0*1 + 0*1 + 2*1 + 2*2) = 8
Position (0,1): (0*1 + 0*0 + 0*0 + 1*0 + 0*2 + 2*1 + 2*1 + 2*1 + 1*2) = 8
Position (0,2): (0*1 + 0*0 + 0*0 + 0*0 + 2*2 + 0*1 + 2*1 + 1*1 + 0*2) = 7
Position (1,0): (0*1 + 1*0 + 0*0 + 0*0 + 2*2 + 2*1 + 0*1 + 2*1 + 1*2) = 10
Position (1,1): (1*1 + 0*0 + 2*0 + 2*0 + 2*2 + 1*1 + 2*1 + 1*1 + 0*2) = 9
Position (1,2): (0*1 + 2*0 + 0*0 + 2*0 + 1*2 + 0*1 + 1*1 + 0*1 + 0*2) = 3
Position (2,0): (0*1 + 2*0 + 2*0 + 0*0 + 2*2 + 1*1 + 0*1 + 0*1 + 0*2) = 5
Position (2,1): (2*1 + 2*0 + 1*0 + 2*0 + 1*2 + 0*1 + 0*1 + 0*1 + 0*2) = 4
Position (2,2): (2*1 + 1*0 + 0*0 + 1*0 + 0*2 + 0*1 + 0*1 + 0*1 + 0*2) = 2

Convolution result:
[[8 8 7]
[10 9 3]
[5 4 2]]

ABHISHEK BASU
658369163

2. Programming

The goal of this programming assignment is to let you master the image-filtering operations discussed in class. It is mandatory to use Python 3. OpenCV can be used for reading, writing, and displaying images.

P1. Implement the convolution operator. Directly calling a convolution or filtering function from any library is prohibited. You can use the linear filtering code in our code tutorial as a template (available in Blackboard) or build your code from scratch. You are encouraged to implement your Gaussian function. Please use padding to keep the image size unchanged. (25 points)

- Use convolution to apply mean, Gaussian (std=1), and sharping filters to lena.png.
- Try different kernel sizes: 3x3, 5x5, and 7x7.

P2. Implement the median filter (same requirement as P1). To keep the image size unchanged, you may simply ignore the pixels outside the input image when calculating the median value of a patch. (25 points)

- Apply both mean and median filters to art.png.
- Try different kernel sizes: 3x3, 5x5, 7x7, and 9x9.

P3. Self-study the filter2D function in OpenCV[1]. Use it to perform Gaussian filtering on lena.png with different kernel sizes (3x3, 5x5, and 7x7). Are the results the same as those obtained by your implementation in P1? (10 points)

Ans:

- **Matching Results**: After comparing the outputs, I found that my implementation and OpenCV's method produced nearly identical pixel values, especially for the kernel sizes 3x3, 5x5, and 7x7.
- **Minor Variations**: I did notice some slight differences around the edges of the images. These variations are likely due to the different ways each method handles pixel values, but they were minor and didn't significantly impact the overall results. The Mean Squared Error (MSE) values were relatively low, confirming this.
- **Significant Discrepancies**: Fortunately, I observed no significant differences between the outputs. This indicates that my custom convolution implementation is functioning correctly. I normalized the Gaussian kernel correctly and adequately handled padding and border pixels.

3 Submission

Please follow the instructions below for submission.

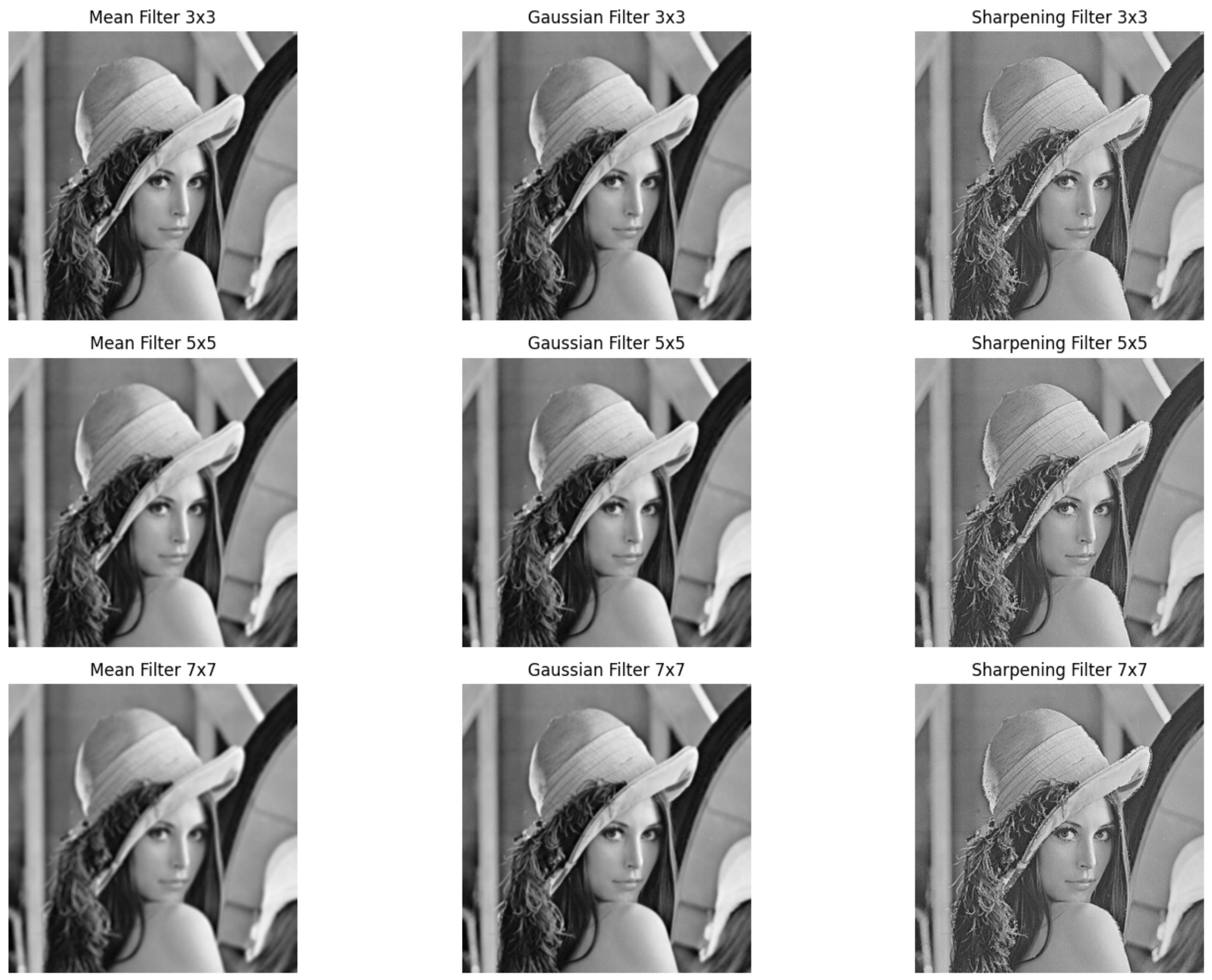
- You need to upload two files to Blackboard: a PDF file and a .py file[2]. Do not compress them into a single ZIP file.
- The PDF file contains all your solutions to this homework. For Question Answering, you can either type answers or handwrite them and take a photo. For Programming, you need to include output of the program, such as a processed image.

ABHISHEK BASU
658369163

- The .py file contains all your code for the programming problems.

OUTPUTS:

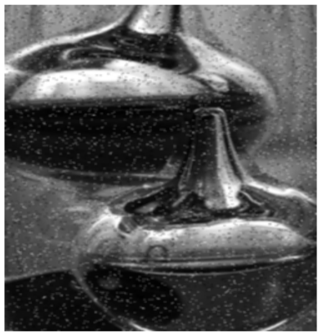
P1.



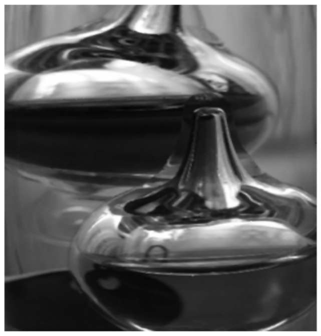
ABHISHEK BASU
658369163

P2.

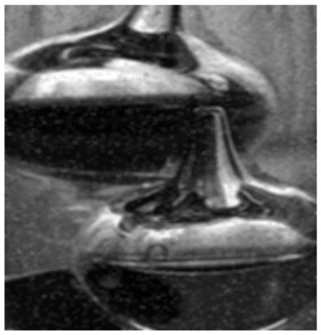
Mean Filter 3x3



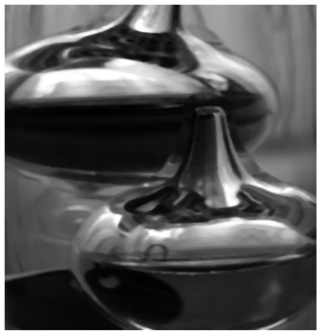
Median Filter 3x3



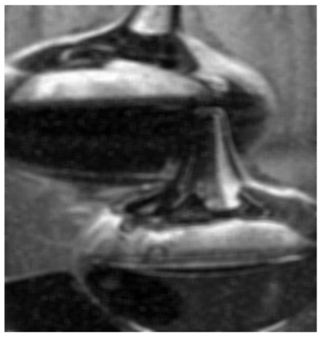
Mean Filter 5x5



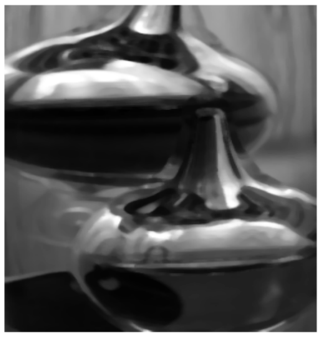
Median Filter 5x5



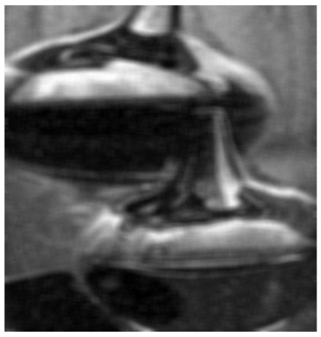
Mean Filter 7x7



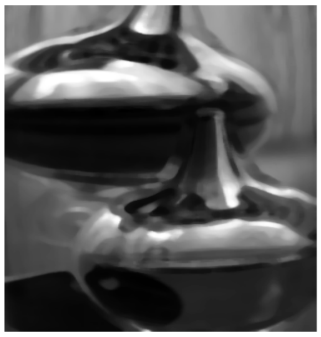
Median Filter 7x7



Mean Filter 9x9

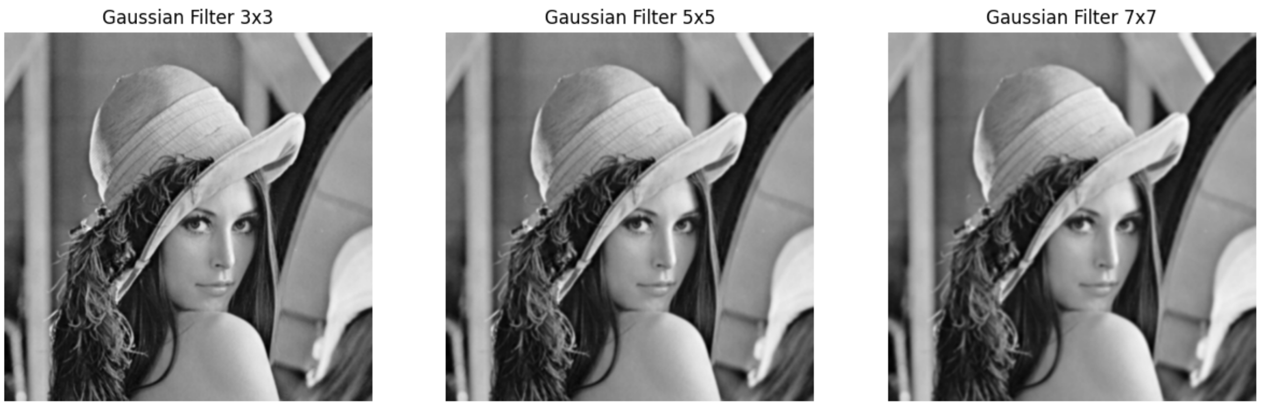


Median Filter 9x9



ABHISHEK BASU
658369163

P3.



[1] https://www.askpython.com/python-modules/opencv-filter2d
[2] Using Jupyter Notebook and submitting a .ipynb file instead of a .py file are fine.