S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 6

Aim: Write programs to apply different filter on given image using OpenCV
i Averaging Filter ii. Gaussian Filter iii. Median Filter iv. Bilateral Filter.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: VII/IV

Academic Session: 2025-2026

Date of Performance:

Date of Submission:

AIM: Write programs to apply different filter on given image using OpenCV

i. Averaging Filter ii. Gaussian Filter iii. Median Filter iv. Bilateral Filter.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To be able to explain the principles of averaging, Gaussian, median, and bilateral filters.
- To be able to apply averaging filter to remove noise from an image.
- To be able to apply Gaussian filter to blur an image.
- To be able to apply median filter to preserve edges while removing noise from an image.
- To be able to apply bilateral filter to preserve edges while smoothing an image.

THEORY:

Neighborhood processing in spatial domain:

To modify one pixel, we consider values of the immediate neighboring pixels also. For this purpose, 3X3, 5X5, or 7X7 neighborhood mask can be considered. An example of a 3X3 mask is shown below.

f(x-1, y-1) f(x-1, y) f(x-1, y+1)

f(x, y-1) f(x, y) f(x, y+1)

f(x+1, y-1) f(x+1, y) f(x+1, y+1)

Low Pass filtering:

It is also known as the smoothing filter. It removes the high-frequency content from the image. It is also used to blur an image. A low pass averaging filter mask is as shown.

1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9

High Pass Filtering:

It eliminates low-frequency regions while retaining or enhancing the high-frequency components. A high pass filtering mask is as shown.

-1/9 -1/9 -1/9

-1/9 8/9 -1/9

-1/9 -1/9 -1/9

2D Convolution (Image Filtering)

As in one-dimensional signals, images also can be filtered with various low-pass filters (LPF), high-pass filters (HPF), etc. LPF helps in removing noise, blurring images, etc. HPF filters help in finding edges in images.

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

OpenCV provides a function cv.filter2D() to convolve a kernel with an image. As an example, we will try an averaging filter on an image. A 5x5 averaging filter kernel will look like the below:

Implementation in OpenCV:

These are the most commonly used OpenCV image smoothing filters-

- Averaging Filter
- Gaussian Filter
- Median Filter
- Bilateral Filter
- **Averaging Filter**
  Average (or mean) filtering is a method of 'smoothing' images by reducing the amount of intensity variation between neighbouring pixels. The average filter works by moving through the image pixel by pixel, replacing each value with the average value of neighbouring pixels, including itself.

The Averaging Filter technique takes the average of all the pixels under the kernel area and replaces the central element. The functions cv2.blur() and cv2.boxFilter() can be used to perform the averaging filter. Both functions smooth an image using the kernel.

Syntax of cv2.blur()

cv2.blur(image, ksize)

Here, the image is the image source, and ksize is the size of blurring kernel.

Syntax of cv2.boxFilter()

cv2.boxFilter(src, dst, depth, ksize, anchor, normalize, bordertype)

In this function , the src is the image source, dst is the destination image of the same size, and depth denotes the output image depth. The anchor denotes the anchor points. By default, it is at the kernel point (cordinate (-1,1)). The ksize is the size of blurring kernel and normalize is the flag which specifies whether the kernel should be normalized or not. The bordertype is an integer object that represents the type of the border used.

PROGRAM CODE:

- **Gaussian Filter**

The Gaussian filter blurs the desired area and cuts the noise with higher frequencies. It works the same as mean filters while representing average weight uniformly. These are linear filters that reduce the noise and blur the edges effectively. They are created as matrices in digital image processing, passing through each pixel of the selected portion.

The OpenCV Gaussian filtering provides the cv2.GaussianBlur() method to blur an image by using a Gaussian Kernel. Each pixel in an image gets multiplied by a Gaussian Kernel. It means, a Gaussian Kernel is a square array of pixels.

Syntax of Gaussian Filter

cv2.GaussianBlur(src, ksize, sigma_x, dst, sigma_y, border_type)
src- the input image,
ksize- Gaussian kernel size (width and height), the width and height can have different values and must be positive and odd,
sigma_x- Gaussian kernel standard deviation along the X-axis,
dst- output image,
sigma_y- Gaussian kernel standard deviation along the Y-axis,
boader_type- image boundaries.

PROGRAM CODE:

- **Median Filter**

Median filter is one of the well-known order-statistic filters due to its good performance for some specific noise types such as "Gaussian," "random," and "salt and pepper" noises. According to the median filter, the center pixel of a M × M neighborhood is replaced by the median value of the corresponding window. Note that noise pixels are considered to be very different from the median.

Using this idea median filter can remove this type of noise problems. We use this filter to remove the noise pixels on the protein crystal images before binarization operation.

Python OpenCV provides the cv2.medianBlur() function to blur the image with a median kernel. This is a non-linear filtering technique. It is highly effective in removing salt-and-pepper noise. This takes the median of all the pixels under the kernel area and replaces the central component with this median value. Since we are taking a middle, the output image will have no new pixel esteems other than that in the input image.

Syntax of Median Filter

cv2.medianBlur(image, ksize)

Here, the image represents the image for operation. The ksize is a size object representing the size of the kernel.

Example of Median Filter

In the given program, we are importing the required modules. Then we are reading the image using the imread() function. Then we are utilizing the medianBlur() function to remove the noise from the image. Then we are displaying the noiseless image as the output on the screen.

PROGRAM CODE:

- **Bilateral Filter**

A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.

This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

Python OpenCV provides the cv2.bilateralFilter() function to blur the image with a bilateral filter. This function can be applied to reduce noise while keeping the edges sharp.

Syntax of Bilateral Filter

cv2.bilateralFilter(image, dst, d, sigmaColor, sigmaSpace)

image- image source,

dst- destination image,

d- specifies the diameter of the pixel neighbourhood,

sigmaColor- specifies the filter sigma in the color space.

sigmaSpace- specifies the filter sigma in the coordinate space.

Program Code:

INPUT & OUTPUT:

Sr. No.	INPUT	OUTPUT
1.		
2.		
3.		
4.		

CONCLUSION:

DISCUSSION QUESTIONS:

1. What are the types of linear and non-linear filters?
2. Explain the differences between an average filter and a median filter.?
3. Which purpose use bilateral filter.?
4. What are the differences between a bilateral filter and a Gaussian filter?
5. Which purpose use Gaussian filter.

REFERENCES:

- Computer Vision: Algorithms and Applications, Richard Szeliski, 2010, Springer.
- Computer Vision - A modern Approach, D. Forsyth, J. Ponce, 2nd Edition 2011, Pearson India.
- OpenCV Computer Vision with Python, Joseph Howse, 2013, Packt Publishing.
- Dictionary of Computer Vision and Image Processing, R. B. Fisher, T. P. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, E. Trucco, C. K. I. Williams, 2nd Edition, 2014, Wiley.
- https://www.geeksforgeeks.org/digital-image-processing-basics/
- https://www.javatpoint.com/digital-image-processing-tutorial
- https://www.simplilearn.com/image-processing-article
- https://www.mygreatlearning.com/blog/digital-image-processing-explained/