Image Understanding and Processing (OpenCv-Python)

Lab Exercise – 06

Year 4 **Semester 1, 2025**

Goal

- Apply custom made filters to images using cv2.filter2D( )
- Apply averaging filter using cv2.blur( ) or cv2.boxFilter( )
- Apply various low pass filters to smooth images (cv2.medianBlur( ) and cv2.GaussianBlur( ))
- Application of image blurring

1. Image filtering using 2D convolution

A Low pass filter helps in removing noise, or blurring the image. OpenCV provides a function, cv2.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 can be defined as follows:

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

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Load the image here

kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(img,-1,kernel)


result = np.hstack((img,dst))
cv2.imshow('result',result)

cv2.waitKey(0)
```

2. Image Averaging using Box Filter

Image averaging is done by convolving the image with a normalized box filter. It simply takes the average of all the pixels under kernel area and replaces the central element with this average. This is done by the function cv2.blur() or cv2.boxFilter().

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Create the code with a kernel of 3x3 size and apply cv2.blur() and cv2.boxFilter() separately.

3. Median Filtering and Gaussian Filtering

The function cv2.medianBlur() computes the median of all the pixels under the kernel window and the central pixel is replaced with this median value. This is highly effective in removing salt-and-pepper noise.

```
median = cv2.medianBlur(img,3)
```

Create the code with a kernel of 3x3 size and apply cv2.medianBlur().

The above code can be modified for Gaussian blurring:

```
blur = cv2.GaussianBlur(img,(11,11),0)
```

Gaussian filtering is highly effective in removing Gaussian noise from the image.

4. Application of image blurring

Complete the code below to obtain the image outputs

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Load the image here


#Create custom kernal of size 15x15 and apply to the input image


dst = cv2.filter2D(img,-1,kernel)

#Apply thresholding operator to highlight the largest object


result = np.hstack((img,dst,thresh))
cv2.imshow('result',result)

cv2.waitKey(0)
```