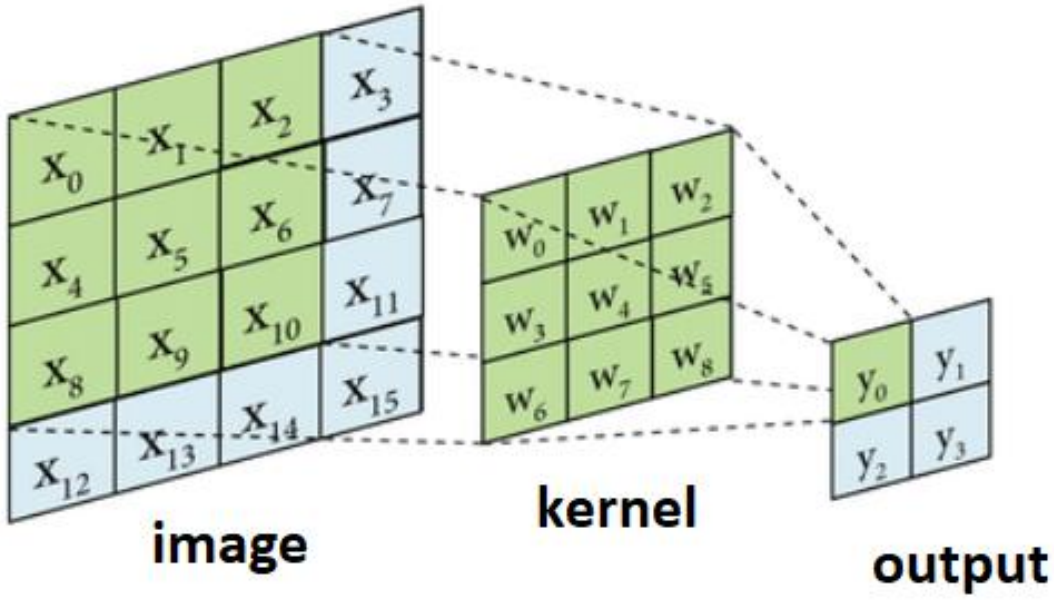
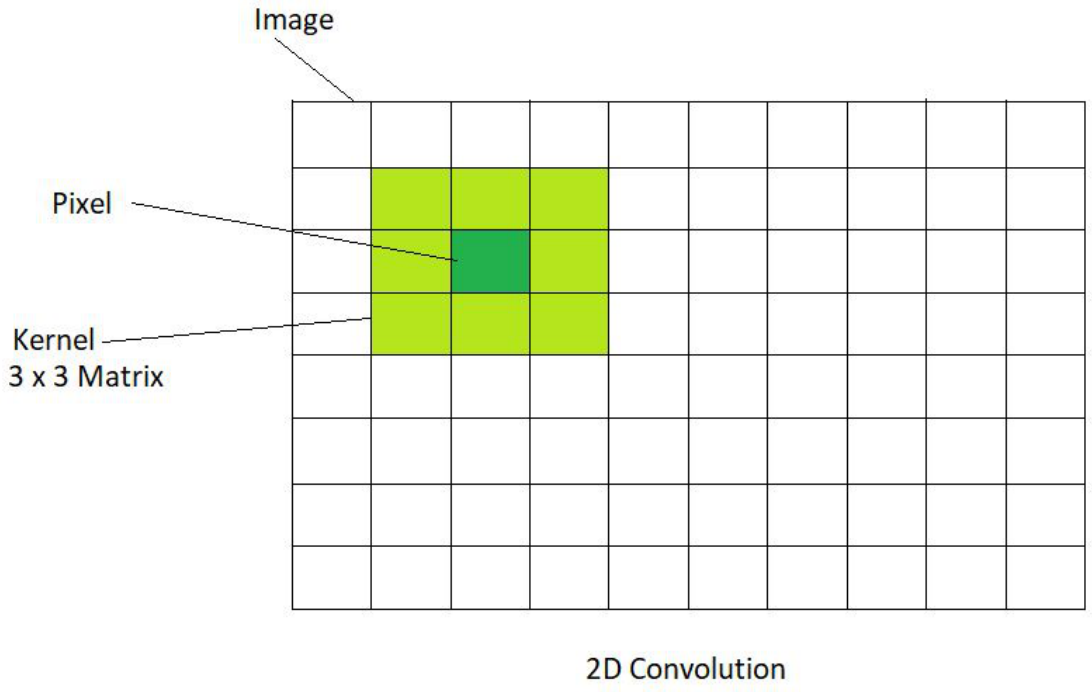
ปฏิบัติการ

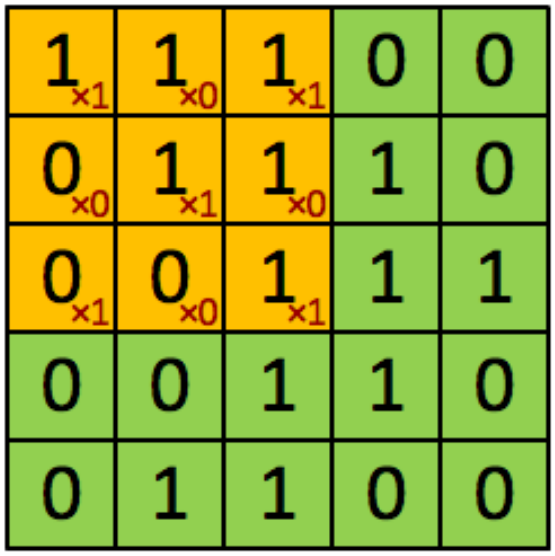
Lab 4 – Convolution, Smoothing, Filters

1. Convolution



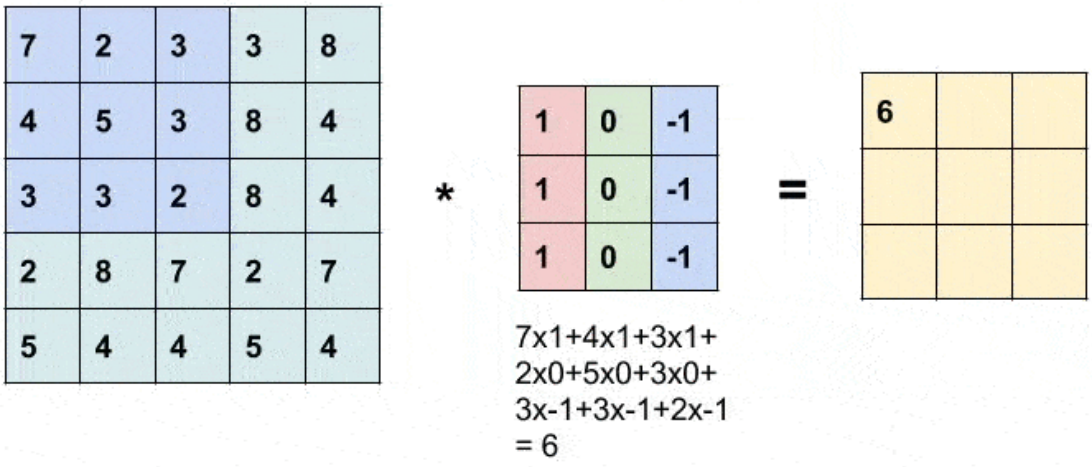


2D Convolution



Image

Convolved Feature




```
import numpy as np


image = np.array([
```

```
    [3, 0, 1, 2, 7, 4],
    [1, 5, 8, 9, 6, 3],
    [2, 7, 2, 5, 1, 8],
    [6, 3, 4, 0, 4, 5],
    [8, 3, 6, 4, 3, 2],
    [3, 7, 9, 2, 8, 3]])

kernel = np.array([
    [-1, -1, -1],
    [-1,  8, -1],
    [-1, -1, -1]])

def convolve2d(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    pad_height = kernel_height // 2
    pad_width = kernel_width // 2

    padded_image = np.zeros((image_height + 2 * pad_height, image_width + 2 * pad_width))
    padded_image[pad_height:-pad_height, pad_width:-pad_width] = image

    output = np.zeros_like(image)

    for i in range(image_height):
        for j in range(image_width):
            region = padded_image[i:i+kernel_height, j:j+kernel_width]
            output[i, j] = np.sum(region * kernel)

    return output

output_image = convolve2d(image, kernel)

print("Original Image:")
print(image)
print("\nKernel:")
```

```
print(kernel)
print("\nOutput Image:")
print(output_image)
```

#1 จงปรับปรุงโค้ดเพื่อให้ตรงกับข้อมูลดังรูปตัวอย่างการทำคอนโวลูชันด้านบน และนำโค้ดที่แก้ไข และผลลัพธ์ที่ได้มาใส่ด้านล่าง

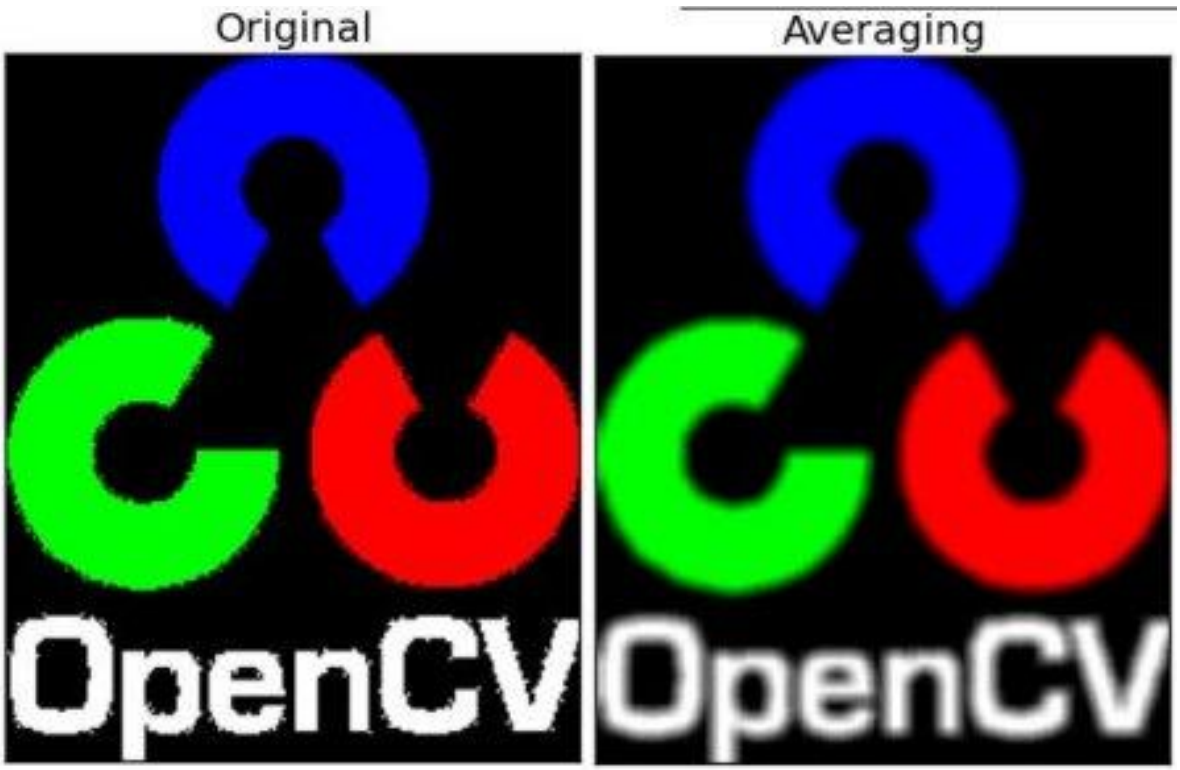
```
import numpy as np

image = np.array([
    [7, 2, 3, 3, 8],
    [4, 5, 3, 8, 4],
    [3, 3, 2, 8, 4],
    [2, 8, 7, 2, 7],
    [5, 4, 4, 5, 4]
])
kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])

def convolve2d_no_padding(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    output = np.zeros((output_height, output_width), dtype=int)

    for i in range(output_height):
        for j in range(output_width):
            region = image[i:i+kernel_height, j:j+kernel_width]
            output[i, j] = np.sum(region * kernel)
    return output
```

```
output_image = convolve2d_no_padding(image, kernel)

print("Original Image:")
print(image)
print("\nKernel:")
print(kernel)
print("\nOutput Image:")
print(output_image)
```

```
.py
Original Image:
[[7 2 3 3 8]
 [4 5 3 8 4]
 [3 3 2 8 4]
 [2 8 7 2 7]
 [5 4 4 5 4]]

Kernel:
[[ 1  0 -1]
 [ 1  0 -1]
 [ 1  0 -1]]

Output Image:
[[ 6 -9 -8]
 [-3 -2 -3]
 [-3  0 -2]]
PS D:\COD_E\001_Project\CLASS_2024\Comvi\Week04>
```

2. Smoothing (Image Blurring)

2.1 Smoothing (Averaging)



$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

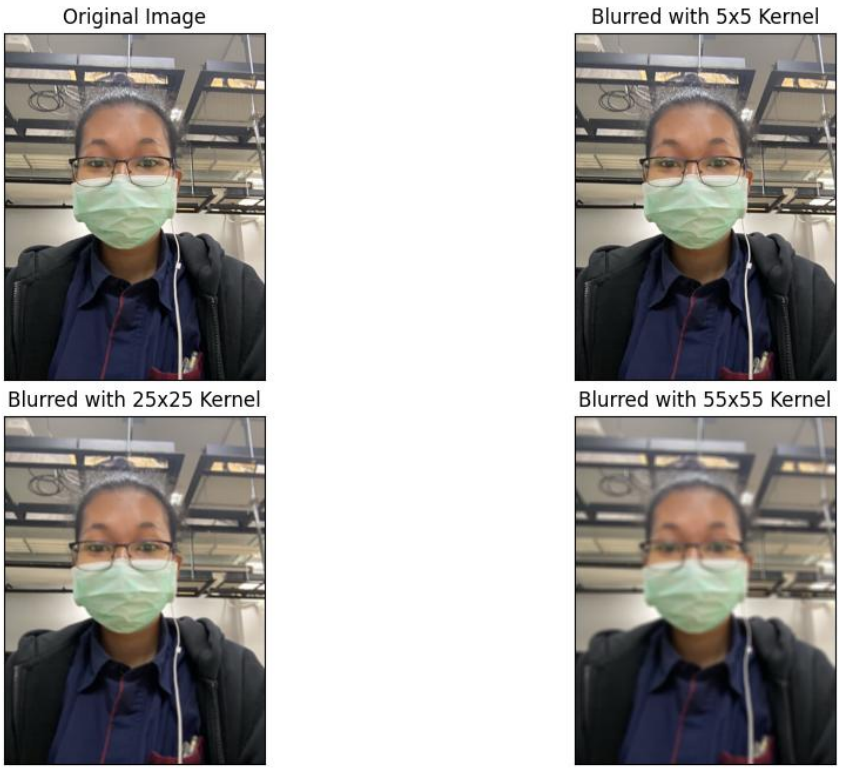
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('ภาพตนเอง')
blur = cv2.blur(img,(5,5)


plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
```

```
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

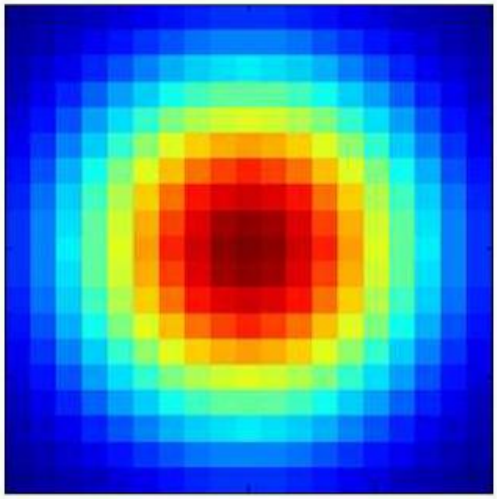
#2 จงแก้ไขโค้ด และแคปภาพผลลัพธ์จากการทำ Smoothing (Averaging) โดยใช้ภาพตัวเองโดยเปลี่ยนแปลงขนาดของหน้ากากที่ [5,5], [25,25], [55,55]

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('D:/COD_E/001_Project/CLASS_2024/Comvi/Week04/me.jpg')

blur_5x5 = cv2.blur(img, (5, 5))        # ขนาดหน้ากาก 5x5
blur_25x25 = cv2.blur(img, (25, 25))    # 25x25
blur_55x55 = cv2.blur(img, (55, 55))    # 55x55

'''เมื่อขนาดหน้ากากใหญ่ขึ้น ภาพจะถูกทำให้เบลอ (blurred) มากขึ้น'''

# แสดงภาพต้นฉบับและภาพที่ทำการ Smoothing
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 2)
plt.imshow(cv2.cvtColor(blur_5x5, cv2.COLOR_BGR2RGB))
plt.title('Blurred with 5x5 Kernel')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 3)
plt.imshow(cv2.cvtColor(blur_25x25, cv2.COLOR_BGR2RGB))
plt.title('Blurred with 25x25 Kernel')
plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 4)
```

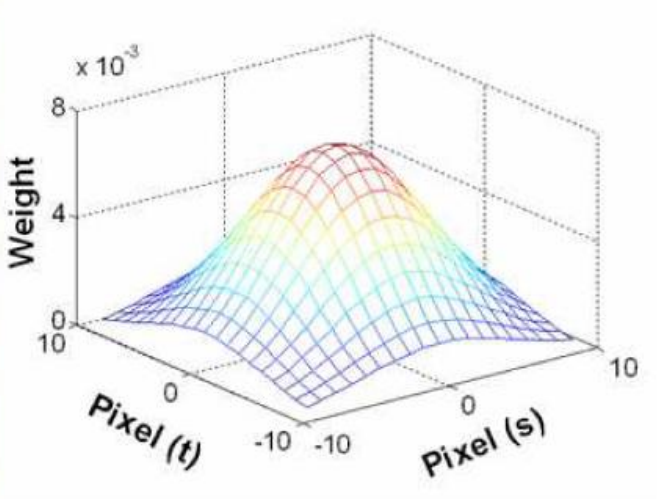
```
plt.imshow(cv2.cvtColor(blur_55x55, cv2.COLOR_BGR2RGB))
plt.title('Blurred with 55x55 Kernel')
plt.xticks([]), plt.yticks([])

plt.tight_layout()
plt.show()
```



2.2 Smoothing (Gaussian Filtering)



(a)



(b)

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('ภาพตนเอง')
blur = cv2.GaussianBlur(img,(5,5),0)
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

#3 จงแก้ไขโค้ดจากข้อที่ 1 และนำหน้ากากแบบ Gaussian ทั้งสามขนาดมาทดสอบกับภาพตัวเอง และแคปผลลัพธ์ที่ได้

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

image = np.array([
    [7, 2, 3, 3, 8],
    [4, 5, 3, 8, 4],
    [3, 3, 2, 8, 4],
    [2, 8, 7, 2, 7],
    [5, 4, 4, 5, 4]
])

kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])

# Convolution function
def convolve2d_no_padding(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    # size calculation (without padding)
    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    # Prepare the output array
    output = np.zeros((output_height, output_width), dtype=int)

    # Perform the convolution
    for i in range(output_height):
        for j in range(output_width):
            region = image[i:i + kernel_height, j:j + kernel_width]
            output[i, j] = np.sum(region * kernel)

    return output

# Perform convolution
output_image = convolve2d_no_padding(image, kernel)
```

```
img = cv2.imread('D:/COD_E/001_Project/CLASS_2024/Comvi/Week04/me.jpg')

# Create a list of kernel sizes for Gaussian Blur
kernel_sizes = [(3, 3), (5, 5), (7, 7)]
blurred_images = []

# Apply Gaussian Blur with different kernel sizes
for kernel_size in kernel_sizes:
    blur = cv2.GaussianBlur(img, kernel_size, 0)
    blurred_images.append(blur)

plt.figure(figsize=(15, 10))

# Plot original image and Gaussian Blurred images
plt.subplot(2, 4, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  # Convert 'BGR -> RGB' for matplotlib
plt.title("Original Gaussian Blur Image")
plt.axis('off')

for i, kernel in enumerate(kernel_sizes):
    plt.subplot(2, 4, i + 2)
    plt.imshow(cv2.cvtColor(blurred_images[i], cv2.COLOR_BGR2RGB))  # Convert BGR -> RGB
    plt.title(f'Gaussian {kernel[0]}, {kernel[1]}')
    plt.axis('off')

plt.tight_layout()
plt.show()
```