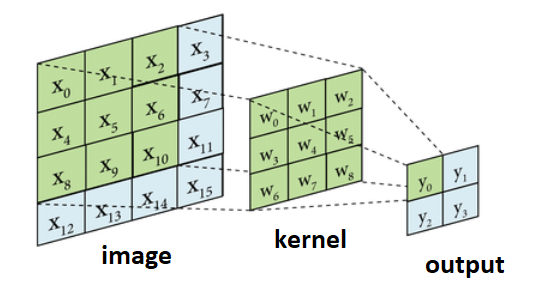
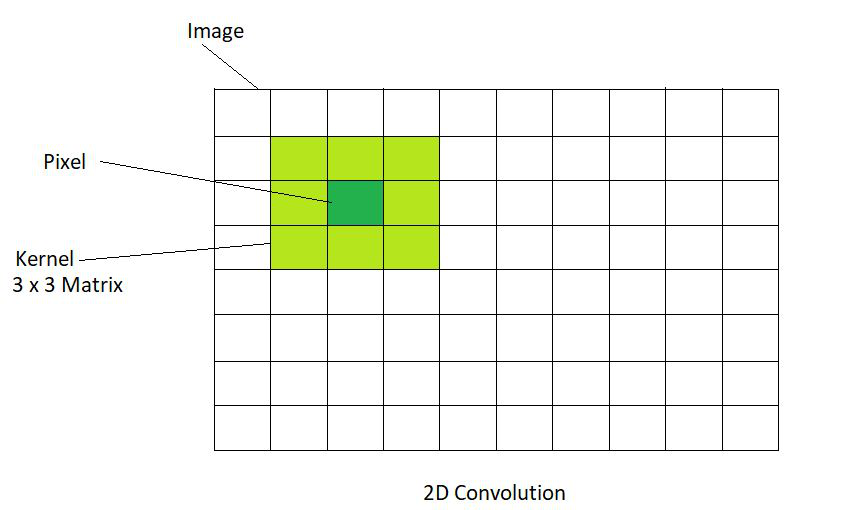
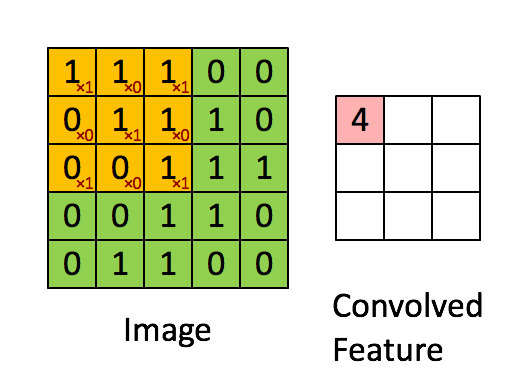
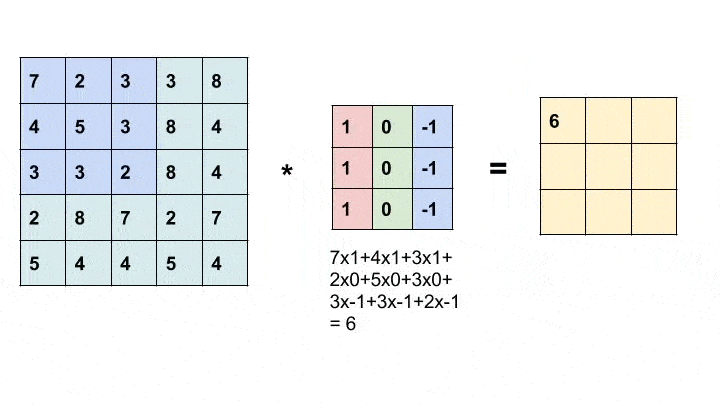
|  |
| --- |
| ปฏิบัติการ  Lab 4 – Convolution, Smoothing, Filters |

1. Convolution









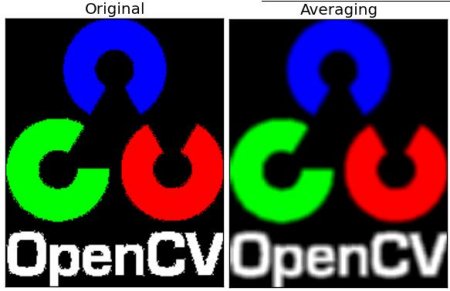
|  |
| --- |
| import numpy as np  image = np.array([  [3, 0, 1, 2, 7, 4],  [1, 5, 8, 9, 6, 3],  [2, 7, 2, 5, 1, 8],  [6, 3, 4, 0, 4, 5],  [8, 3, 6, 4, 3, 2],  [3, 7, 9, 2, 8, 3]])  kernel = np.array([  [-1, -1, -1],  [-1, 8, -1],  [-1, -1, -1]])  def convolve2d(image, kernel):  kernel\_height, kernel\_width = 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) |

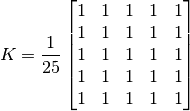
#1 จงปรับปรุงโค้ดเพื่อให้ตรงกับข้อมูลดังรูปตัวอย่างการทำคอนโวลูชันด้านบน และนำโค้ดที่แก้ไข และผลลัพธ์ที่ได้มาใส่ด้านล่าง

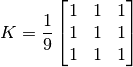
|  |
| --- |
| import numpy as np  image = np.array([      [7, 2, 3, 3, 8],      [4, 5, 3, 8, 4],      [3, 3, 2, 8, 4],      [2, 8, 7, 2, 7],      [5, 4, 4, 5, 4]  ])  kernel = np.array([      [1, 0, -1],      [1, 0, -1],      [1, 0, -1]  ])  def convolve2d\_no\_padding(image, kernel):      kernel\_height, 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) |

**2. Smoothing (Image Blurring)**

**2.1 Smoothing (Averaging)**

****



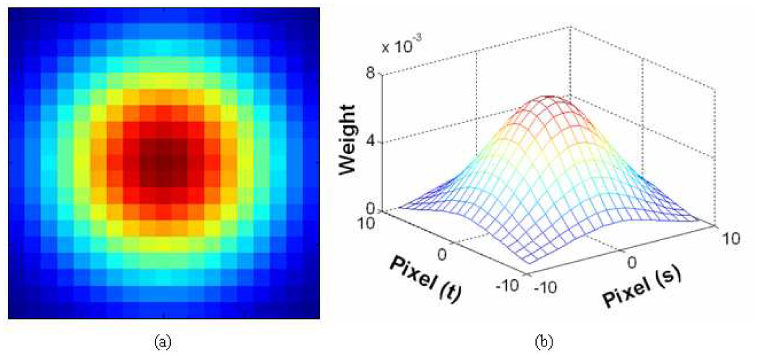


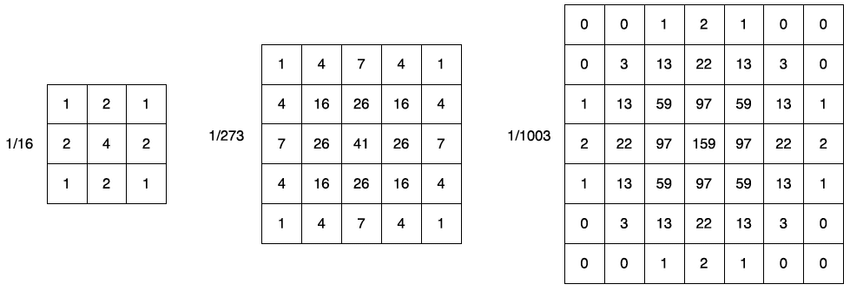
|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  img = cv2.imread('ภาพตนเอง')  blur = cv2.blur(img,(5,5)  plt.subplot(121),plt.imshow(img),plt.title('Original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(blur),plt.title('Blurred')  plt.xticks([]),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





|  |
| --- |
| 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() |