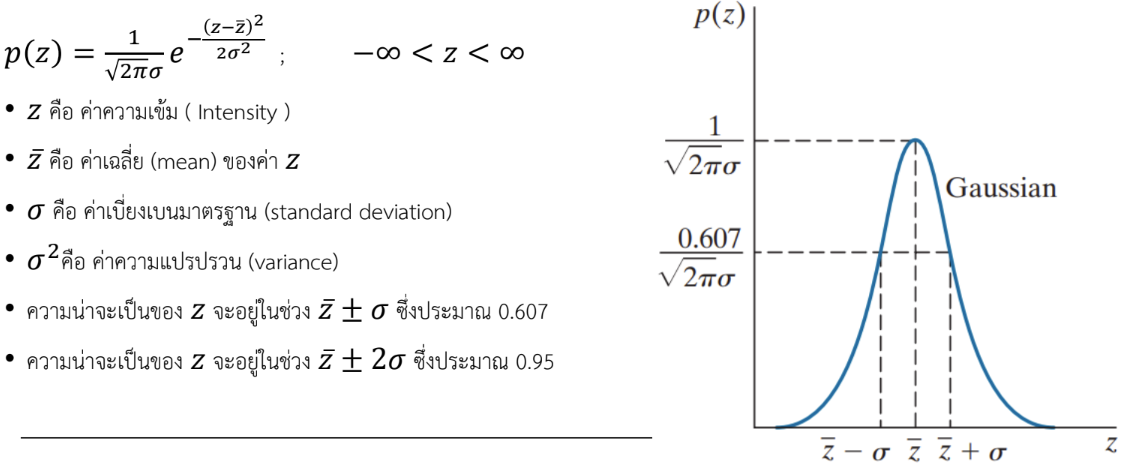
|  |
| --- |
| ปฏิบัติการ  Lab 6 – Noise |

1. Gaussian Noise

เป็น noise ที่พบบ่อยในภําพและสัญญําณ Model ถูกสร้ํางขึ้นจํากกระบวนกํารสุ่มตํามกํารแจกแจง Gaussian



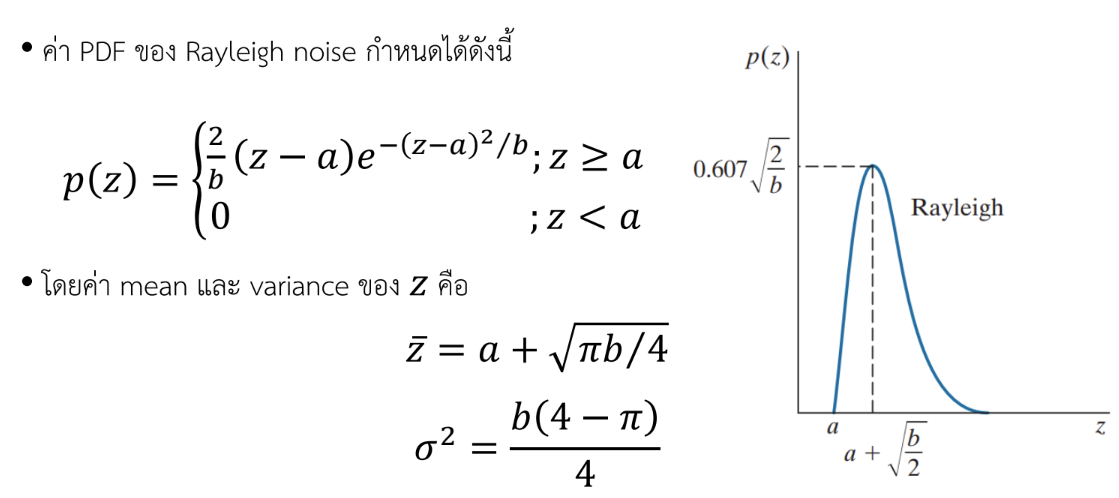
|  |
| --- |
| import numpy as np  def add\_gaussian\_noise(image, mean=0, sigma=5):  gaussian\_noise = np.random.normal(mean, sigma, image.shape)  gaussian\_noise = gaussian\_noise.reshape(image.shape)  noisy\_image = image + gaussian\_noise  noisy\_image = np.clip(noisy\_image, 0, 255)  noisy\_image = noisy\_image.astype(np.uint8)  return noisy\_image  import cv2  import matplotlib.pyplot as plt  img = cv2.imread("test.jpg",0)  ret = add\_gaussian\_noise(img)  plt.figure(figsize=(5, 5))  plt.subplot(1, 1, 1)  plt.title("Original Image")  plt.imshow(img, cmap="gray", interpolation='nearest', aspect='auto')  plt.show() |

โจทย์ที่ #1 จงแก้ไขโค้ดข้างต้นเพื่อนำฟังก์ชัน add\_gaussian\_noise() มาใช้งาน โดยกำหนดให้ทดลองสุ่มค่า mean และ sigma และอธิบายว่าพารามิเตอร์ mean และ sigma มีผลอย่างไร? (ลองทดสอบเปลี่ยนค่า mean และ sigma และนำมาแสดงผลร่วมกับภาพต้นฉบับ โดยเพิ่มมาอีกประมาณ 3 ภาพ)

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import cv2  def add\_gaussian\_noise(image, mean=0, sigma=25):      row, col, ch = image.shape      gauss = np.random.normal(mean, sigma, (row, col, ch))      noisy = np.clip(image + gauss, 0, 255)      return noisy.astype(np.uint8)  image = cv2.imread('bb.jpg')  image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  mean\_values = [0, 10, 50]  sigma\_values = [10, 25, 50]  fig, axs = plt.subplots(len(mean\_values), len(sigma\_values), figsize=(15, 10))  axs[0, 0].imshow(image)  axs[0, 0].set\_title("Original Image")  axs[0, 0].axis('off')  index = 1  for i, mean in enumerate(mean\_values):      for j, sigma in enumerate(sigma\_values):          noisy\_image = add\_gaussian\_noise(image, mean, sigma)          axs[i, j].imshow(noisy\_image)          axs[i, j].set\_title(f"Mean={mean}, Sigma={sigma}")          axs[i, j].axis('off')  plt.tight\_layout()  plt.show() |

2. Rayleigh noise

เป็น noise ที่พบบ่อยในภําพและสัญญําณ Model ถูกสร้ํางขึ้นจํากกระบวนกํารสุ่มตํามกํารแจกแจง Gaussian

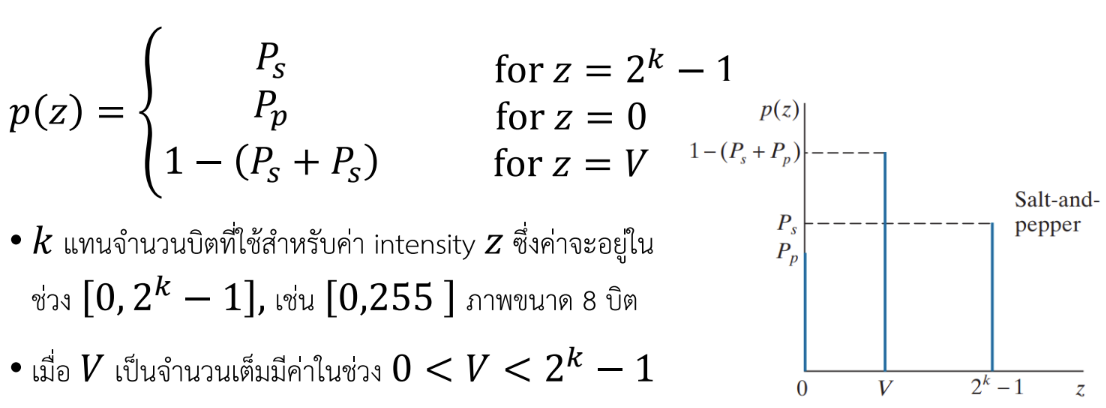


|  |
| --- |
| import numpy as np  import cv2  img = cv2.imread('test.jpg',cv2.IMREAD\_GRAYSCALE)  image = img.astype(np.float64)  noise\_std = 0.2  noise = np.random.rayleigh(noise\_std, img.shape)  noisy\_image = cv2.addWeighted(image, 1, noise, 70, 0.0).astype(np.uint8)  cv2.imshow('Image', img)  cv2.imshow('Noise', noise)  cv2.imshow('Noisy Image', noisy\_image)  cv2.waitKey(0) |

โจทย์ที่ #2 จงทดลองใช้ Rayleigh noise โดยกำหนดให้ทดลองสุ่มค่า noise\_std และอธิบายว่าพารามิเตอร์ noise\_std มีผลอย่างไร และโค้ดข้างต้นมีความสัมพันธ์กับสมการตามทฤษฎีอย่างไร?

|  |
| --- |
| ค่า noise\_std สูงขึ้น ก็จะทำให้ความเข้ม noise มีค่าสูงขึ้น  การกระจายตัวของ noise กว้างขึ้น และภาพมีความผิดเพี้ยนมากขึ้น    import numpy as np  import cv2  img = cv2.imread('bb.jpg', cv2.IMREAD\_GRAYSCALE)  image = img.astype(np.float64)  noise\_std = 0.2  noise = np.random.rayleigh(noise\_std, img.shape)  noisy\_image = cv2.addWeighted(image, 1, noise, 70, 0.0).astype(np.uint8)  width = 500  height = 400  dim = (width, height)  resized\_img = cv2.resize(img, dim, interpolation=cv2.INTER\_AREA)  resized\_noise = cv2.resize(noise, dim, interpolation=cv2.INTER\_AREA)  resized\_noisy = cv2.resize(noisy\_image, dim, interpolation=cv2.INTER\_AREA)  cv2.imshow('Image', resized\_img)  cv2.imshow('Noise', resized\_noise)  cv2.imshow('Noisy Image', resized\_noisy)  cv2.waitKey(0) |

3. Salt-and-Pepper Noise



|  |
| --- |
| Algorithm AddSaltAndPepperNoise(image, salt\_probability, pepper\_probability):  Input:  image - a 2D or 3D matrix representing the grayscale or color image  salt\_probability - probability of adding white pixels (salt noise)  pepper\_probability - probability of adding black pixels (pepper noise)  Output:  noisy\_image - the image with added Salt-and-Pepper noise    Step 1: Initialize noisy\_image as a copy of the original image  noisy\_image = Copy(image)    Step 2: Get image dimensions  rows, cols = Dimensions(image)  Step 3: Iterate over each pixel in the image  For i = 1 to rows:  For j = 1 to cols:  Step 4: Generate a random number r between 0 and 1  r = Random(0, 1)  Step 5: Check the probability thresholds  If r < salt\_probability:  noisy\_image[i, j] = MaximumPixelValue # Add salt (white pixel)  Else if r < salt\_probability + pepper\_probability:  noisy\_image[i, j] = MinimumPixelValue # Add pepper (black pixel)  Else:  noisy\_image[i, j] = image[i, j] # Keep original pixel  Step 6: Return the noisy image  Return noisy\_image |

โจทย์ที่ #3 จงใช้ pseudo code ข้างต้น เพื่อนำไปเขียนโค้ดสำหรับการใส่ Salt-and-Pepper Noise ลงไปในภาพ?

|  |
| --- |
| import numpy as np  import cv2  import matplotlib.pyplot as plt  def add\_salt\_and\_pepper\_noise(image, salt\_probability, pepper\_probability):      noisy\_image = np.copy(image)      rows, cols = image.shape[:2]      for i in range(rows):          for j in range(cols):              r = np.random.random()              if r < salt\_probability:                  noisy\_image[i, j] = 255              elif r < salt\_probability + pepper\_probability:                  noisy\_image[i, j] = 0      return noisy\_image  image = cv2.imread('bb.jpg', cv2.IMREAD\_GRAYSCALE)  salt\_probability = 0.02  pepper\_probability = 0.02  noisy\_image = add\_salt\_and\_pepper\_noise(image, salt\_probability, pepper\_probability)  plt.figure(figsize=(10, 5))  plt.subplot(1, 2, 1)  plt.imshow(image, cmap='gray')  plt.title('Original Image')  plt.axis('off')  plt.subplot(1, 2, 2)  plt.imshow(noisy\_image, cmap='gray')  plt.title('Image with Salt-and-Pepper Noise')  plt.axis('off')  plt.tight\_layout()  plt.show() |