ข้อ 1

Smoothing filters

a. ) Averaging filter

def averaging\_filter(img, mask\_n=3):

    mask = np.ones([mask\_n,mask\_n], dtype=np.float32)

    mask = mask / (mask\_n\*\*2)

    filtered\_img = np.zeros\_like(img)

    m,n = img.shape

    offset = mask\_n // 2

    for i in range(offset, m-offset):

        for j in range(offset, n-offset):

            mask\_area = img[i-offset:i+offset+1, j-offset:j+offset+1]

            filtered\_img[i, j]= np.sum(mask\_area\*mask)

    return filtered\_img

img1 = cv2.imread("noisy\_img1.jpg", cv2.IMREAD\_GRAYSCALE)

img2 = cv2.imread("noisy\_img2.jpg", cv2.IMREAD\_GRAYSCALE)

output\_folder = "./smoothing\_filters"

if not os.path.exists(output\_folder):

    os.makedirs(output\_folder)

avg\_filtered\_1  = averaging\_filter(img1, 3)

cv2.imwrite(os.path.join(output\_folder, "noisy1\_avg\_filtered.jpg"), avg\_filtered\_1)

avg\_filtered\_2  = averaging\_filter(img2, 3)

cv2.imwrite(os.path.join(output\_folder, "noisy2\_avg\_filtered.jpg"), avg\_filtered\_2)

ภาพต้นฉบับ



noisy\_img1 noisy\_img2



average noisy\_img1 average noisy\_img2

อธิบายเพิ่มเติม : ภาพที่ได้จากการทำ average filtering จะทำให้ภาพมีลักษณะที่เนียนขึ้นหรือที่เรียกว่าเป็น smoothing filters โดยหลักการคือเอาค่าที่อยู่ใน mask มาเฉลี่ยๆกัน ทำให้ภาพผลลัพธ์ที่ได้มีค่าออกมาประมาณบริเวณ pixel ที่พิจารณา โดยจะมีข้อเสีย เช่น รูปที่ 1 ที่มี noise เป็นจุดบนหลังคาที่มีจุดสีดำใกล้ๆกันบริเวณเดียวกัน ทำให้ค่าเฉลี่ยที่ได้ก็จะค่อนไปทางสีดำ

b.) Median filter

def median\_filter(img, mask\_n=3):

    mask = np.ones([mask\_n,mask\_n], dtype=np.float32)

    filtered\_img = np.zeros\_like(img)

    m,n = img.shape

    offset = mask\_n // 2

    for i in range(offset, m-offset):

        for j in range(offset, n-offset):

            mask\_area = img[i-offset:i+offset+1, j-offset:j+offset+1]

            mask\_flatten = (mask\_area\*mask).flatten()

            filtered\_img[i, j]= np.median(mask\_flatten)

    return filtered\_img

img1 = cv2.imread("noisy\_img1.jpg", cv2.IMREAD\_GRAYSCALE)

img2 = cv2.imread("noisy\_img2.jpg", cv2.IMREAD\_GRAYSCALE)

output\_folder = "./smoothing\_filters"

if not os.path.exists(output\_folder):

    os.makedirs(output\_folder)

med\_filtered\_1 = median\_filter(img1,3)

cv2.imwrite(os.path.join(output\_folder, "noisy1\_med\_filtered.jpg"), med\_filtered\_1)

med\_filtered\_2 = median\_filter(img2,3)

cv2.imwrite(os.path.join(output\_folder, "noisy2\_med\_filtered.jpg"), med\_filtered\_2)

ภาพต้นฉบับ

A small cabin in the woods

Description automatically generatedA small cabin in the woods

Description automatically generated

noisy\_img1 noisy\_img2



median noisy\_img1 median noisy\_img2

อธิบายเพิ่มเติม : ภาพที่ได้จากการทำ median filtering จะทำให้ภาพมีลักษณะที่เนียนขึ้นหรือที่เรียกว่าเป็น smoothing filters โดยหลักการเอาค่าที่อยู่ใน mask มาเรียงแล้วเอาค่าที่อยู่กลาง ทำให้ภาพผลลัพธ์ที่ได้มีค่าออกมาเป็นค่าที่น่าเชื่อถือได้ เพราะ noise ที่อยู่ขอบๆ เช่น salt and pepper noise จะถูกตัดออกไป

ข้อ 1

Sharpening filters

a.) Laplacian filter

def laplacian\_filter(img):

    mask\_n = 3

    mask1 = np.array([[ 0,  1,  0],

                      [ 1, -4 , 1],

                      [ 0,  1,  0]])

    mask2 = np.array([[ 1,  1,  1],

                      [ 1, -8,  1],

                      [ 1,  1,  1]])

    laplacian1 = np.zeros\_like(img, dtype=np.int16)

    laplacian2 = np.zeros\_like(img, dtype=np.int16)

    m, n = img.shape

    offset = mask\_n // 2

    for i in range(offset, m-offset):

        for j in range(offset, n-offset):

            mask\_area = img[i-offset:i+offset+1, j-offset:j+offset+1]

            laplacian1[i, j] = np.sum(mask\_area \* mask1)

            laplacian2[i, j] = np.sum(mask\_area \* mask2)

    return laplacian1, laplacian2

def enhance\_with\_laplacian(img):

    laplacian1, laplacian2 = laplacian\_filter(img)

    subtracted\_img\_1 = img.astype(np.int16) - laplacian1

    subtracted\_img\_2 = img.astype(np.int16) - laplacian2

    clipped\_img\_1 = np.clip(subtracted\_img\_1, 0, 255).astype(np.uint8)

    clipped\_img\_2 = np.clip(subtracted\_img\_2, 0, 255).astype(np.uint8)

    return clipped\_img\_1, clipped\_img\_2

img = cv2.imread("blurred\_image.jpg", cv2.IMREAD\_GRAYSCALE)

output\_folder = "./sharpening\_filters"

if not os.path.exists(output\_folder):

    os.makedirs(output\_folder)

laplacian\_filtered\_1, laplacian\_filtered\_2  = enhance\_with\_laplacian(img)

cv2.imwrite(os.path.join(output\_folder, "laplacian\_filtered\_4-neighbor.jpg"), laplacian\_filtered\_1)

cv2.imwrite(os.path.join(output\_folder, "laplacian\_filtered\_8-neighbor.jpg"), laplacian\_filtered\_2)

ภาพต้นฉบับ

A close up of a plant

Description automatically generated

A close-up of water droplets on a blade of grass

Description automatically generated

Laplacian filter 4 neighbors Laplacian filter 8 neighbors

อธิบายเพิ่มเติม : ภาพที่ได้จาก Laplacian filter จะมีลักษณะที่คมขึ้นหรือที่เรียกว่าเป็น sharpening filter ซึ่งหลักการคือการหาขอบแบบ second-order derivative ซึ่งค่าที่ได้จาก Laplacian filter จะได้ค่าเป็นขอบๆของวัตถุ หลังจากได้ขอบก็จะเอาค่านี้ไปรวมกับภาพเดิมทำให้ภาพเดิมเห็นขอบได้ชัดมากขึ้น

b.) Gradient filter

def sobel\_filter(img):

    mask\_n = 3

    mask\_Gx = np.array([[-1,  0, 1],

                        [-2,  0, 2],

                        [-1,  0, 1]])

    mask\_Gy = np.array([[-1, -2, -1],

                        [ 0,  0,  0],

                        [ 1,  2,  1]])

    filtered\_img\_Gx = np.zeros\_like(img, dtype=np.float64)

    filtered\_img\_Gy = np.zeros\_like(img, dtype=np.float64)

    m, n = img.shape

    offset = mask\_n // 2

    for i in range(offset, m - offset):

        for j in range(offset, n - offset):

            mask\_area = img[i - offset:i + offset + 1, j - offset:j + offset + 1]

            filtered\_img\_Gx[i, j] = np.sum(mask\_area \* mask\_Gx)

            filtered\_img\_Gy[i, j] = np.sum(mask\_area \* mask\_Gy)

    return filtered\_img\_Gx, filtered\_img\_Gy

def enhance\_with\_sobel(img):

    filtered\_Gx, filtered\_Gy = sobel\_filter(img)

    gradient\_magnitude = np.sqrt(filtered\_Gx\*\*2 + filtered\_Gy\*\*2)

    gradient\_magnitude = gradient\_magnitude / np.max(gradient\_magnitude) \* 255

    sharpened\_img = img + gradient\_magnitude

    sharpened\_img = np.clip(sharpened\_img, 0, 255)

    return sharpened\_img.astype(np.uint8)

img = cv2.imread("blurred\_image.jpg", cv2.IMREAD\_GRAYSCALE)

output\_folder = "./sharpening\_filters"

if not os.path.exists(output\_folder):

    os.makedirs(output\_folder)

sobel\_filtered = enhance\_with\_sobel(img)

cv2.imwrite(os.path.join(output\_folder, "sobel\_filtered.jpg"), sobel\_filtered)

ภาพต้นฉบับ

Water droplets on a blade of grass

Description automatically generatedA close up of a plant

Description automatically generated



Sobel filter

อธิบายเพิ่มเติม : การใช้ sobel filter เป็นการหาค่า gradient หรือค่าการเปลี่ยนแปลง ซึ่งค่าที่ได้ออกมาที่จะแตกต่างกับบริเวณอื่นก็จะอยู่บริเวณขอบๆ ใน mask ซึ่งจะมี matrix 2 แบบเหมือนในโค้ด โดยจะมี matrix ที่หา gradient ในแนวแกน  
x และมี matrix ที่หา gradient แนวแกน y เมื่อได้แล้วก็จะนำค่าที่ได้มาหาขนาด คือ เสร็จแล้วก็นำมารวมกับภาพต้นฉบับก็จะได้รูปภาพที่มีการเน้นให้ขอบเห็นได้ชัดเจนมากยิ่งขึ้น