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

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

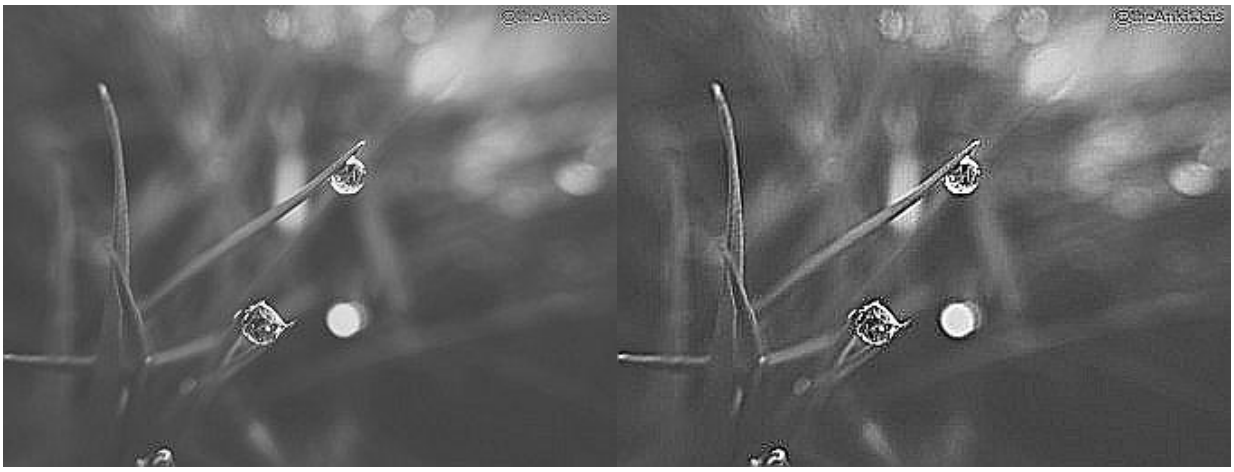
Sharpening filters

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



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

```python
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

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 $\sqrt{G_x^2 + G_y^2}$ เสร็จแล้วก็นำมารวมกับภาพต้นฉบับก็จะได้รูปภาพที่มีการเน้นให้ขอบเห็นได้ชัดเจนมากยิ่งขึ้น