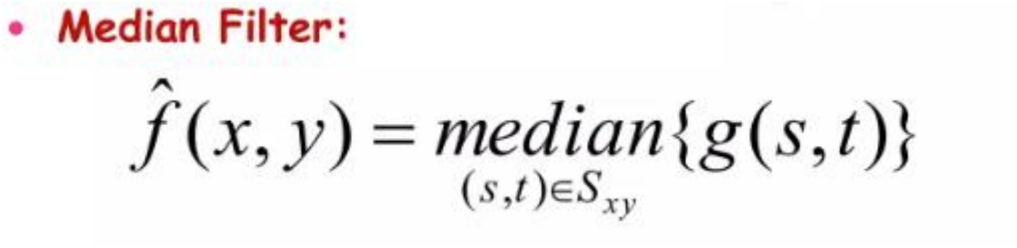
ปฏิบัติการ

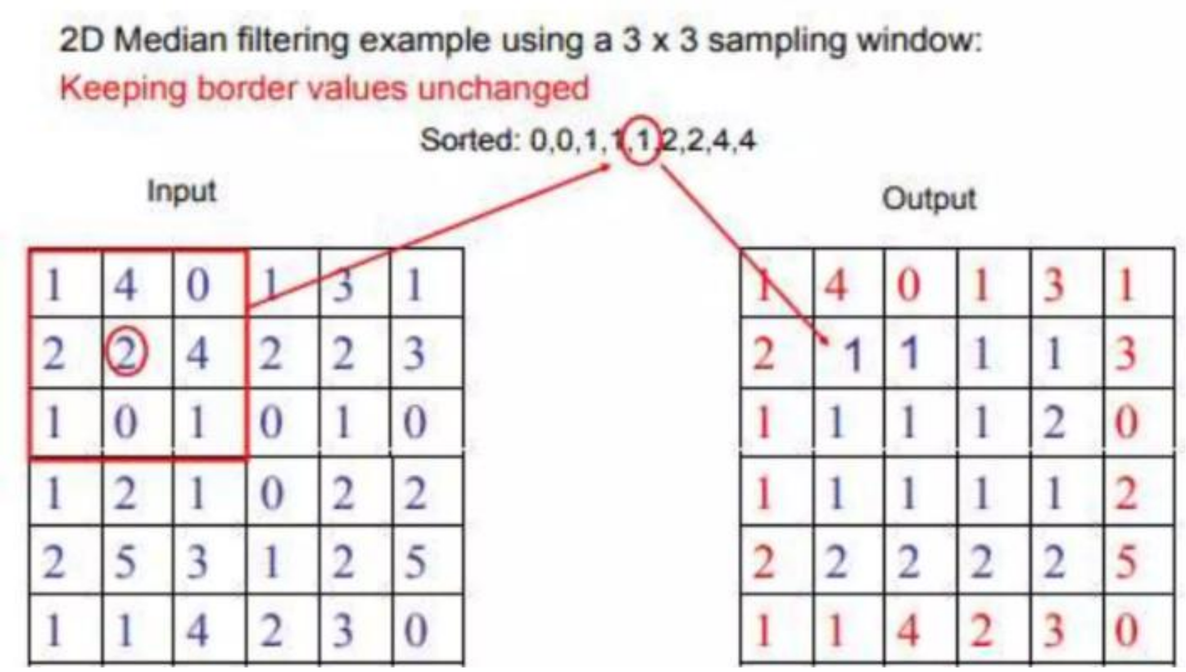
Lab 5 – Order-statistics filters, Sharpening

1. Order-statistics filters



- **Median Filter:**

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$



2D Median filtering example using a 3 x 3 sampling window:

Keeping border values unchanged

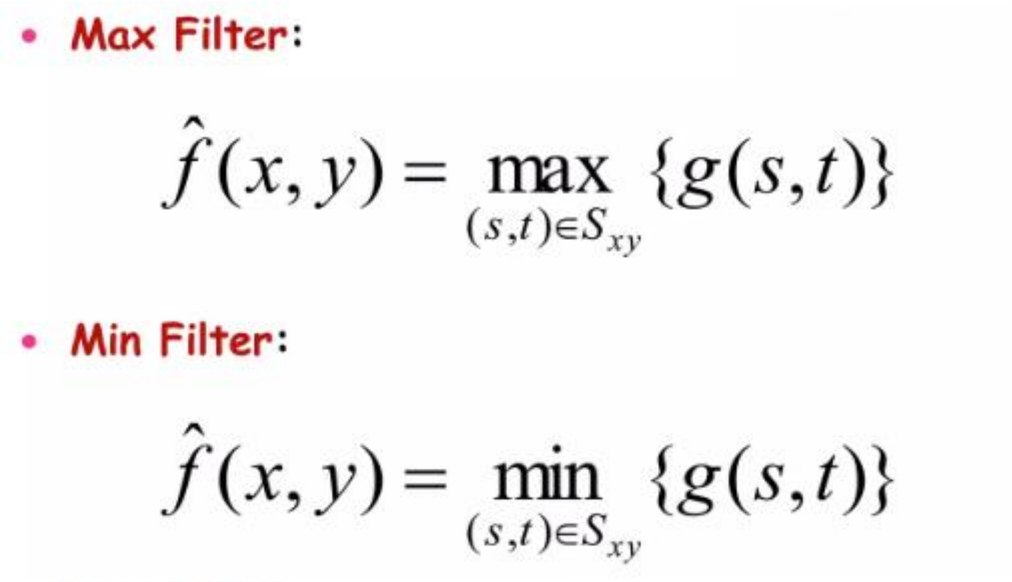
Sorted: 0,0,1,1,1,2,2,4,4

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

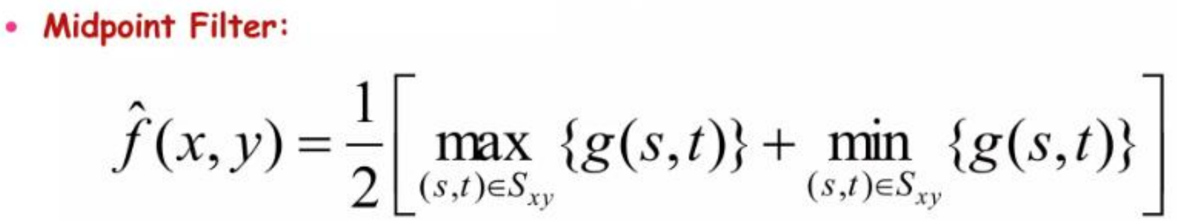


- **Max Filter:**

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

- **Min Filter:**

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$



- **Midpoint Filter:**

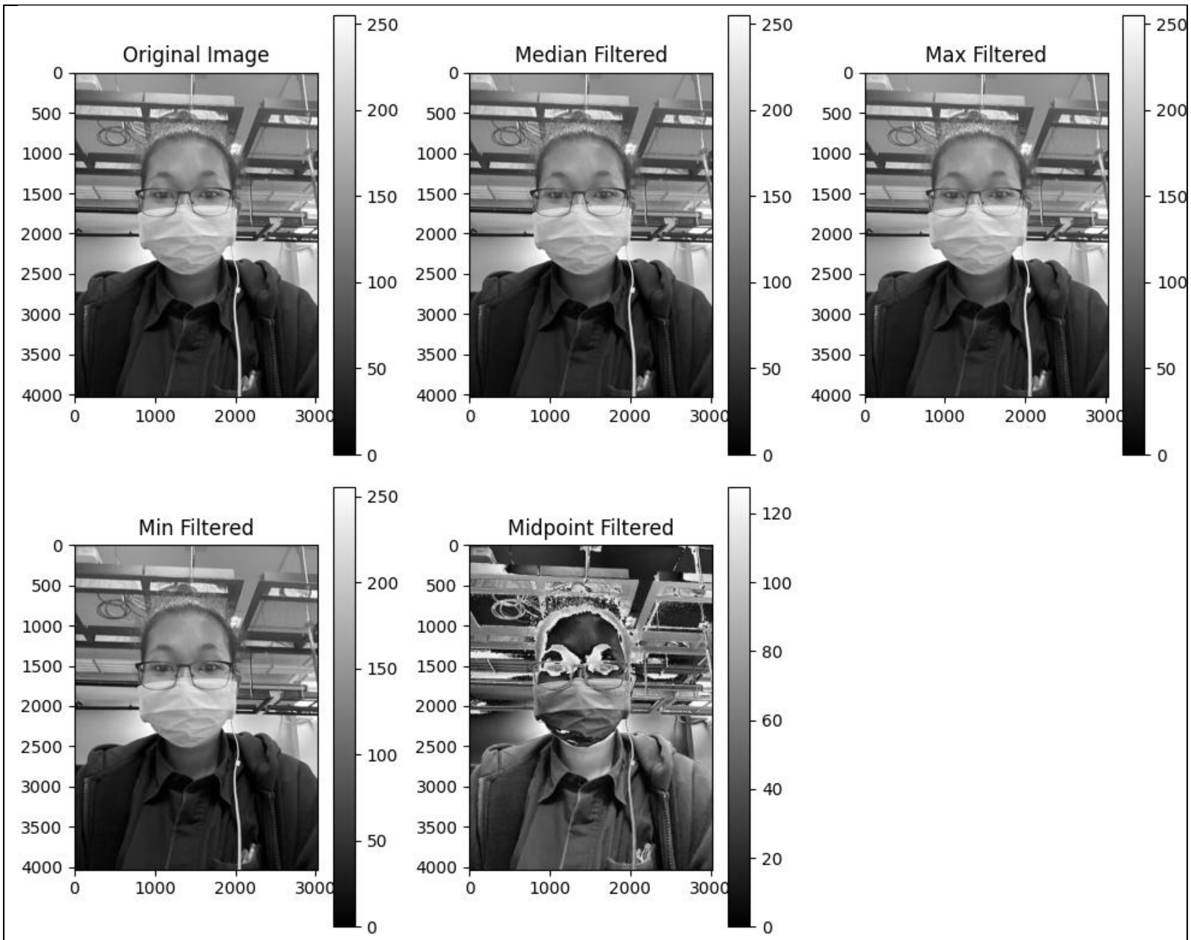
$$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\}\right]$$

```
import numpy as np
from scipy.ndimage import median_filter, maximum_filter, minimum_filter
import matplotlib.pyplot as plt

def apply_median_filter(image, size):
    return median_filter(image, size=size)

def apply_max_filter(image, size):
    return maximum_filter(image, size=size)

def apply_min_filter(image, size):
    return minimum_filter(image, size=size)

# Example usage
def main():
    np.random.seed(0)
    image = np.random.randint(0, 256, (10, 10), dtype=np.uint8)
    print("Original Image:\n", image)
    filter_size = 3
    median_filtered = apply_median_filter(image, size=filter_size)
    max_filtered = apply_max_filter(image, size=filter_size)
    min_filtered = apply_min_filter(image, size=filter_size)

    print("\nMedian Filtered Image:\n", median_filtered)
    print("\nMax Filtered Image:\n", max_filtered)
    print("\nMin Filtered Image:\n", min_filtered)

    plt.figure(figsize=(10, 8))

    plt.subplot(2, 2, 1)
    plt.title("Original Image")
    plt.imshow(image, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 2, 2)
    plt.title("Median Filtered")
    plt.imshow(median_filtered, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 2, 3)
    plt.title("Max Filtered")
    plt.imshow(max_filtered, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 2, 4)
    plt.title("Min Filtered")
    plt.imshow(min_filtered, cmap="gray")
    plt.colorbar()

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

#1 จงปรับปรุงโค้ดเพื่อนำ Midpoint filter เพิ่มมาอีก 1 อัลกอริทึม และปรับปรุงให้โค้ดทำงานร่วมกับรูปภาพของตนเอง และปรับปรุงการแสดงผลให้แสดงภาพผลลัพธ์แทนอาเรย์ จากนั้นนำโค้ดที่แก้ไข และผลลัพธ์ที่ได้มาใส่ด้านล่าง

```
def apply_filter(image, filter_size, mode):
    def get_window(x, y):
        window = []
        for i in range(-half_size, half_size + 1):
            for j in range(-half_size, half_size + 1):
                xi = min(max(x + i, 0), rows - 1)
                yj = min(max(y + j, 0), cols - 1)
                window.append(image[xi][yj])
        return window

    # Validate inputs
    if filter_size % 2 == 0:
        raise ValueError("Filter size must be an odd integer.")

    half_size = filter_size // 2
    rows, cols = len(image), len(image[0])
    filtered_image = [[0 for _ in range(cols)] for _ in range(rows)]

    for x in range(rows):
        for y in range(cols):
            window = get_window(x, y)

            if mode == 'median':
                filtered_image[x][y] = sorted(window)[len(window) // 2]
            elif mode == 'max':
                filtered_image[x][y] = max(window)
            elif mode == 'min':
                filtered_image[x][y] = min(window)
            else:
                raise ValueError("Mode must be 'median', 'max', or 'min'.")

    return filtered_image
```

```
if __name__ == "__main__":
    input_image = [
        [1, 2, 3, 4, 5],
        [6, 7, 8, 9, 10],
        [11, 12, 13, 14, 15],
        [16, 17, 18, 19, 20],
        [21, 22, 23, 24, 25]
    ]

    filter_size = 3

    print("Original Image:")
    for row in input_image:
        print(row)

    median_filtered = apply_filter(input_image, filter_size, 'median')
    print("\nMedian Filtered Image:")
    for row in median_filtered:
        print(row)

    max_filtered = apply_filter(input_image, filter_size, 'max')
    print("\nMax Filtered Image:")
    for row in max_filtered:
        print(row)

    min_filtered = apply_filter(input_image, filter_size, 'min')
    print("\nMin Filtered Image:")
    for row in min_filtered:
        print(row)
```

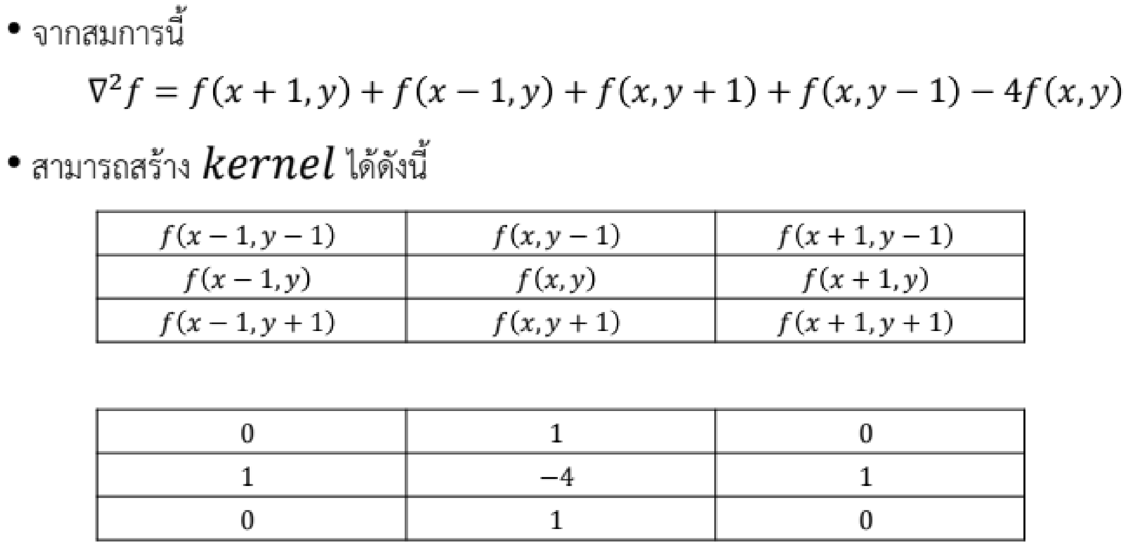


```
import numpy as np
from scipy.ndimage import median_filter, maximum_filter, minimum_filter
import matplotlib.pyplot as plt
from PIL import Image

def apply_median_filter(image, size):
    return median_filter(image, size=size)

def apply_max_filter(image, size):
    return maximum_filter(image, size=size)

def apply_min_filter(image, size):
    return minimum_filter(image, size=size)

def apply_midpoint_filter(image, size):
    min_filtered = minimum_filter(image, size=size)
    max_filtered = maximum_filter(image, size=size)
    return (min_filtered + max_filtered) / 2
```

```
def main():
    image = np.array(Image.open('me.jpg').convert('L'))
    filter_size = 3

    # Apply filters
    median_filtered = apply_median_filter(image, size=filter_size)
    max_filtered = apply_max_filter(image, size=filter_size)
    min_filtered = apply_min_filter(image, size=filter_size)
    midpoint_filtered = apply_midpoint_filter(image, size=filter_size)

    # Display results
    plt.figure(figsize=(10, 8))

    plt.subplot(2, 3, 1)
    plt.title("Original Image")
    plt.imshow(image, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 3, 2)
    plt.title("Median Filtered")
    plt.imshow(median_filtered, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 3, 3)
    plt.title("Max Filtered")
    plt.imshow(max_filtered, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 3, 4)
    plt.title("Min Filtered")
    plt.imshow(min_filtered, cmap="gray")
    plt.colorbar()

    plt.subplot(2, 3, 5)
    plt.title("Midpoint Filtered")
    plt.imshow(midpoint_filtered, cmap="gray")
    plt.colorbar()
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

2. Sharpening

2.1 Laplacian



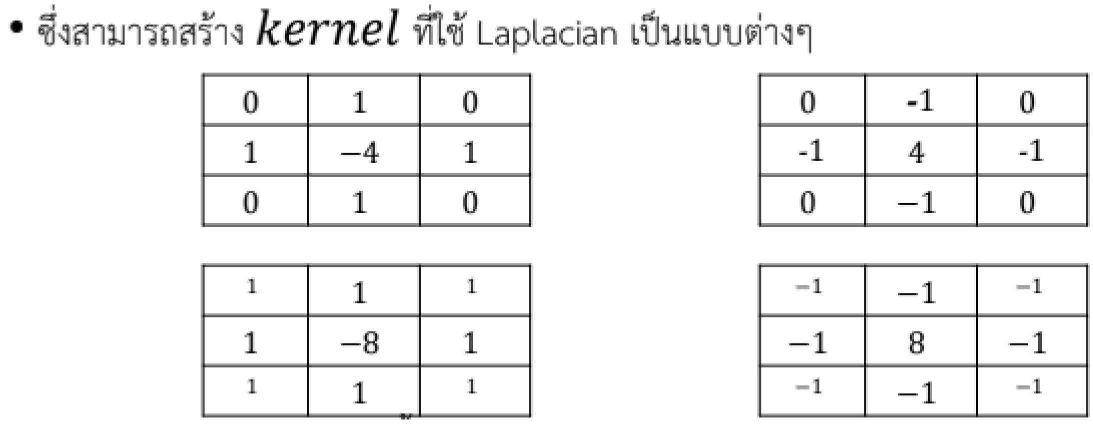
- จากสมการนี้

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

- สามารถสร้าง *kernel* ได้ดังนี้

$f(x-1,y-1)$	$f(x,y-1)$	$f(x+1,y-1)$
$f(x-1,y)$	$f(x,y)$	$f(x+1,y)$
$f(x-1,y+1)$	$f(x,y+1)$	$f(x+1,y+1)$

0	1	0
1	−4	1
0	1	0



- ซึ่งสามารถสร้าง *kernel* ที่ใช้ Laplacian เป็นแบบต่างๆ

0	1	0
1	−4	1
0	1	0

0	-1	0
-1	4	-1
0	−1	0

1	1	1
1	−8	1
1	1	1

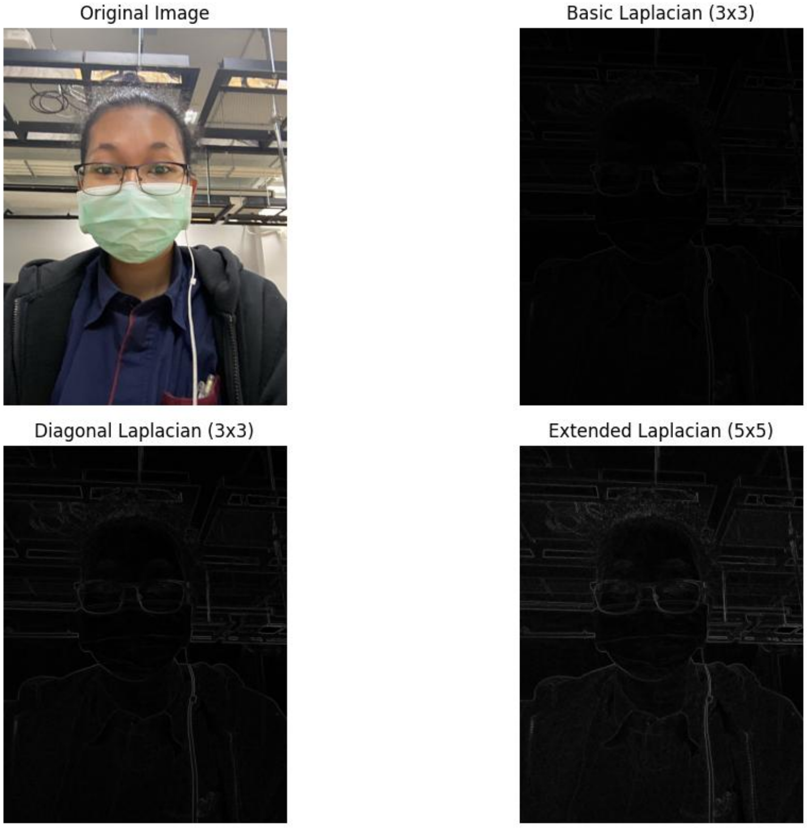
−1	−1	−1
−1	8	−1
−1	−1	−1

```
def apply_laplacian_sharpening(image):
    kernel = [
        [0, -1, 0],
        [-1, 4, -1],
        [0, -1, 0]
    ]
    rows, cols = len(image), len(image[0])
    sharpened_image = [[0 for _ in range(cols)] for _ in range(rows)]

    def get_pixel(x, y):
        if x < 0 or x >= rows or y < 0 or y >= cols:
            return 0
        return image[x][y]

    for x in range(rows):
        for y in range(cols):
            laplacian_sum = 0
            for i in range(-1, 2):
                for j in range(-1, 2):
                    laplacian_sum += kernel[i + 1][j + 1] * get_pixel(x + i, y + j)
            sharpened_image[x][y] = min(max(image[x][y] + laplacian_sum, 0), 255)
    return sharpened_image

if __name__ == "__main__":
    input_image = [
        [10, 20, 30, 40, 50],
        [60, 70, 80, 90, 100],
        [110, 120, 130, 140, 150],
        [160, 170, 180, 190, 200],
        [210, 220, 230, 240, 250]
    ]
    print("Original Image:")
    for row in input_image:
        print(row)
    sharpened_image = apply_laplacian_sharpening(input_image)
    print("\nSharpened Image:")
    for row in sharpened_image:
        print(row)
```

#2 จงแก้ไขโค้ด และแคปภาพผลลัพธ์จากการทำ Laplacian โดยใช้ภาพตัวเองและเปลี่ยนแปลงขนาดของหน้ากากของ Laplacian ให้ครบทุกแบบ

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


def apply_laplacian(image, kernel_type):
    # Convert image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Define different Laplacian kernels
    kernels = {
        'basic': np.array([[0, 1, 0],
                           [1, -4, 1],
                           [0, 1, 0]]),

        'diagonal': np.array([[1, 1, 1],
                              [1, -8, 1],
                              [1, 1, 1]]),

        'extended': np.array([[0, 0, -1, 0, 0],
                              [0, -1, -2, -1, 0],
                              [-1, -2, 16, -2, -1],
                              [0, -1, -2, -1, 0],
                              [0, 0, -1, 0, 0]])
    }

    # Apply custom kernel
    laplacian = cv2.filter2D(gray, -1, kernels[kernel_type])

    # Normalize output
    laplacian = cv2.normalize(laplacian, None, 0, 255, cv2.NORM_MINMAX)

    return laplacian.astype(np.uint8)

# Read your image
image = cv2.imread('me.jpg')

# Apply different Laplacian kernels
basic_laplacian = apply_laplacian(image, 'basic')
diagonal_laplacian = apply_laplacian(image, 'diagonal')
extended_laplacian = apply_laplacian(image, 'extended')
```

```
# Display results
plt.figure(figsize=(12, 8))

plt.subplot(221)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

plt.subplot(222)
plt.imshow(basic_laplacian, cmap='gray')
plt.title('Basic Laplacian (3x3)')
plt.axis('off')

plt.subplot(223)
plt.imshow(diagonal_laplacian, cmap='gray')
plt.title('Diagonal Laplacian (3x3)')
plt.axis('off')

plt.subplot(224)
plt.imshow(extended_laplacian, cmap='gray')
plt.title('Extended Laplacian (5x5)')
plt.axis('off')

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



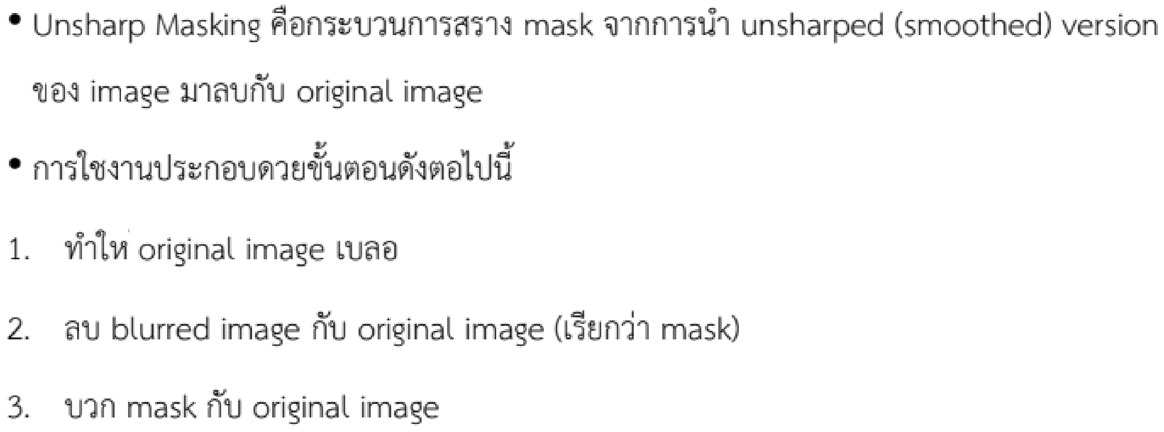
Original Image

Basic Laplacian (3x3)

Diagonal Laplacian (3x3)

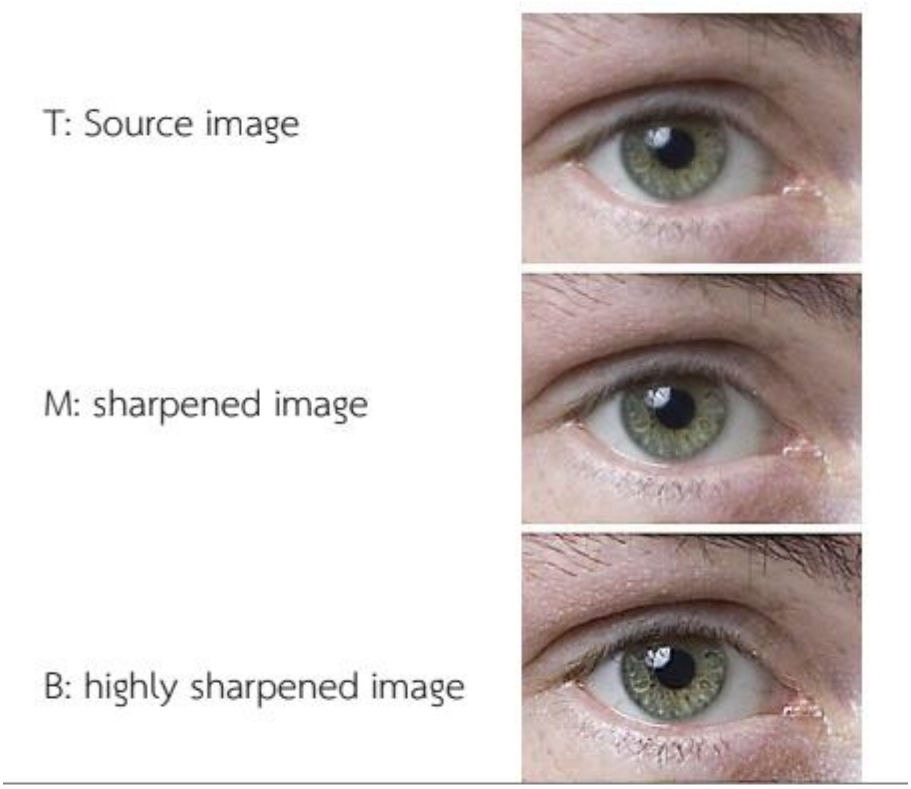
Extended Laplacian (5x5)

2.2 Unsharp masking และ Highboost Filtering



- Unsharp Masking คือกระบวนการสราง mask จากการนำ unsharped (smoothed) version ของ image มาลบกับ original image
- การใชงานประกอบดวยขั้นตอนดังตอไปนี้

1. ทำให original image เบลอ
2. ลบ blurred image กับ original image (เรียกว่า mask)
3. บวก mask กับ original image

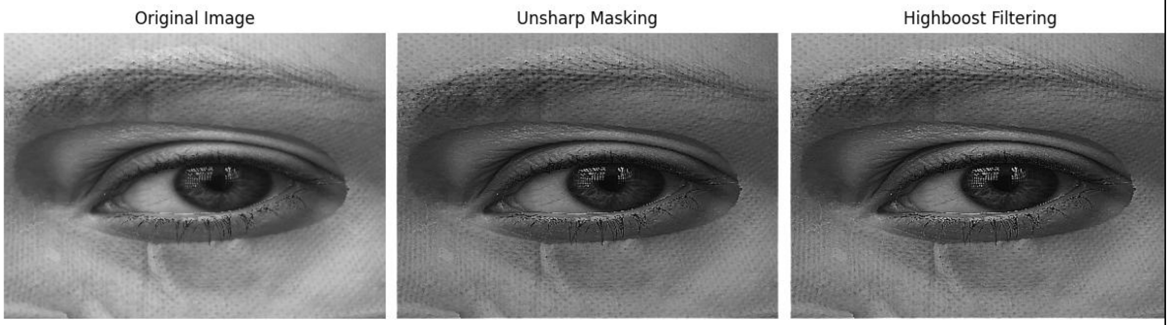


ตัวอย่างการทำ Unsharp masking และ Highboost Filtering

```
import numpy as np


def apply_filter(image, kernel):
    height, width = image.shape
    kernel_size = kernel.shape[0]
    pad = kernel_size // 2
    padded_image = np.pad(image, pad, mode='constant', constant_values=0)
    output = np.zeros_like(image, dtype=np.float32)

    # Convolution operation
    for i in range(height):
        for j in range(width):
            region = padded_image[i:i + kernel_size, j:j + kernel_size]
            output[i, j] = np.sum(region * kernel)

    return output

def unsharp_masking(image, kernel_size=3, alpha=1.5):
    kernel = np.ones((kernel_size, kernel_size)) / (kernel_size ** 2)
    blurred_image = apply_filter(image, kernel)
    mask = image - blurred_image
    sharpened_image = image + alpha * mask

    return sharpened_image

def highboost_filtering(image, kernel_size=3, k=1.5):
    kernel = np.ones((kernel_size, kernel_size)) / (kernel_size ** 2)
    blurred_image = apply_filter(image, kernel)
    mask = image - blurred_image
    highboost_image = image + k * mask

    return highboost_image

if __name__ == "__main__":
    image = np.array([
        [50, 50, 50, 50, 50],
        [50, 100, 100, 100, 50],
        [50, 100, 150, 100, 50],
        [50, 100, 100, 100, 50],
        [50, 50, 50, 50, 50]
    ], dtype=np.float32)
    print("Original Image:")
    print(image)

    sharpened = unsharp_masking(image, kernel_size=3, alpha=1.5)
    print("\nSharpened Image (Unsharp Masking):")
    print(sharpened)

    highboost = highboost_filtering(image, kernel_size=3, k=2.0)
    print("\nHighboost Filtered Image:")
    print(highboost)
```

#3 จงหาตัวอย่างภาพที่นำมาทดสอบกับการทำ Unsharp masking และ Highboost Filteringของโค้ดตัวอย่างเพื่อให้เห็นผลลัพธ์ที่ชัดเจน และแคปภาพผลลัพธ์มาใส่ในคำตอบด้านล่าง [แก้ไขโค้ดให้แสดงภาพด้วย]

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

def apply_filter(image, kernel):
    height, width = image.shape
    kernel_size = kernel.shape[0]
    pad = kernel_size // 2
    padded_image = np.pad(image, pad, mode='constant', constant_values=0)
    output = np.zeros_like(image, dtype=np.float32)

    for i in range(height):
        for j in range(width):
            region = padded_image[i:i + kernel_size, j:j + kernel_size]
            output[i, j] = np.sum(region * kernel)

    return output

def unsharp_masking(image, kernel_size=3, alpha=1.5):
    kernel = np.ones((kernel_size, kernel_size)) / (kernel_size ** 2)
    blurred_image = apply_filter(image, kernel)
    mask = image - blurred_image
    sharpened_image = image + alpha * mask
    return np.clip(sharpened_image, 0, 255)

def highboost_filtering(image, kernel_size=3, k=1.5):
    kernel = np.ones((kernel_size, kernel_size)) / (kernel_size ** 2)
    blurred_image = apply_filter(image, kernel)
    mask = image - blurred_image
    highboost_image = image + k * mask
    return np.clip(highboost_image, 0, 255)

# Read and process image
image = cv2.imread("eye.jpg", cv2.IMREAD_GRAYSCALE)
image = cv2.resize(image, (400, 300))

# Apply filters
sharpened = unsharp_masking(image, kernel_size=3, alpha=1.5)
highboost = highboost_filtering(image, kernel_size=3, k=2.0)

# Display results
plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(132)
plt.imshow(sharpened, cmap='gray')
plt.title('Unsharp Masking')
plt.axis('off')

plt.subplot(133)
plt.imshow(highboost, cmap='gray')
plt.title('Highboost Filtering')
plt.axis('off')
plt.tight_layout()
plt.show()
```


Original Image
Unsharp Masking
Highboost Filtering