Program Soal-Soal UAS

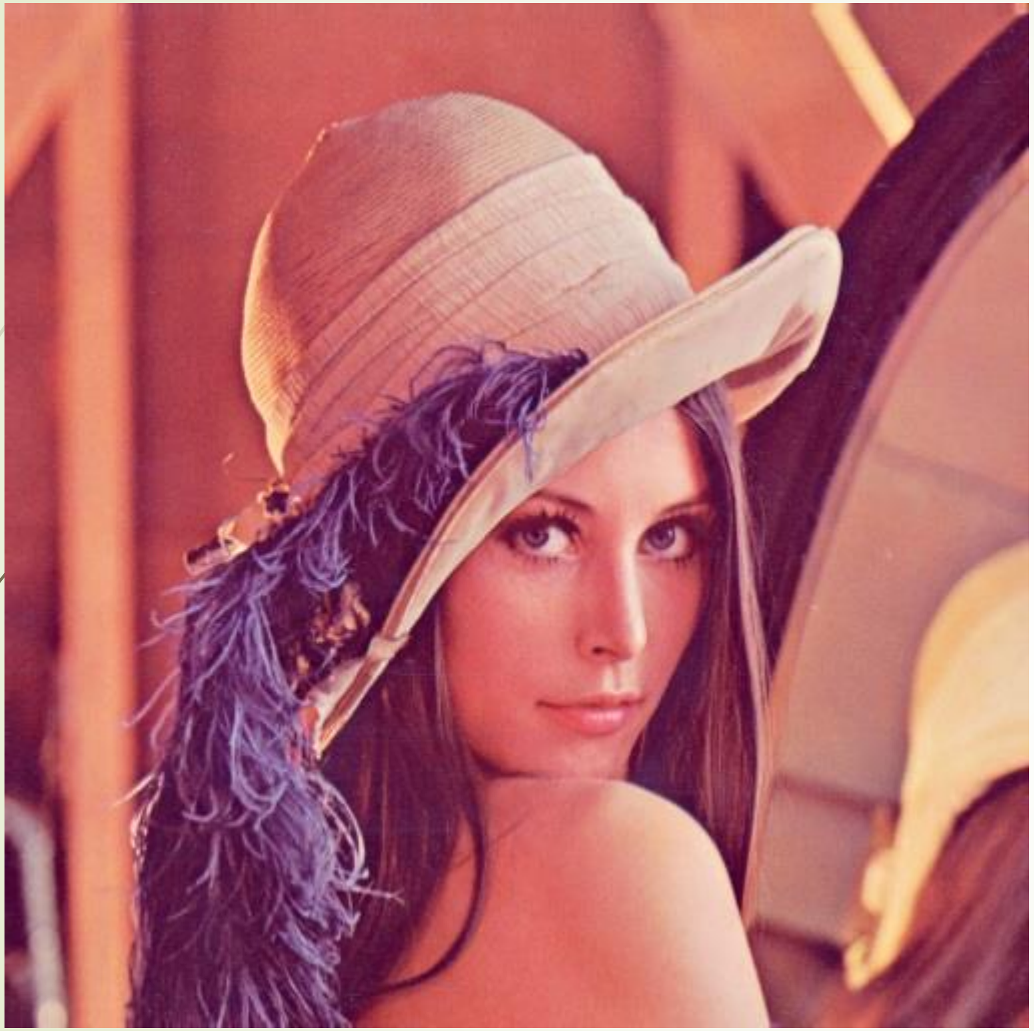
Firnanda Pristiana Nurmaida - 1122800004

Program Template Matching Menggunakan SAD

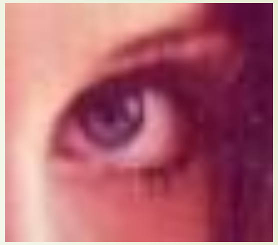
```
import cv2
import numpy as np

def template_matching_sad_high_accuracy(image, template):
    # Konversi gambar ke grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray_template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

    # Hasil korelasi
    result = cv2.matchTemplate(gray_image, gray_template, cv2.TM_SQDIFF)

    # Dapatkan lokasi dengan nilai minimum
    min_val, _, min_loc, _ = cv2.minMaxLoc(result)

    return min_loc
```

```
# Baca gambar dan template
image = cv2.imread('gambar_utama.jpg')  # Ganti 'gambar_utama.jpg' dengan nama gambar yang kamu gunakan
template = cv2.imread('template.jpg')   # Ganti 'template.jpg' dengan nama template yang kamu gunakan

# Panggil fungsi template_matching_sad_high_accuracy
matching_loc = template_matching_sad_high_accuracy(image, template)

# Gambar hasil
result_image = image.copy()
cv2.rectangle(result_image, matching_loc, (matching_loc[0] + template.shape[1], matching_loc[1] + template.shape[0]), (0, 255, 0), 2)

# Tampilkan gambar hasil
cv2.imshow('Hasil', result_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

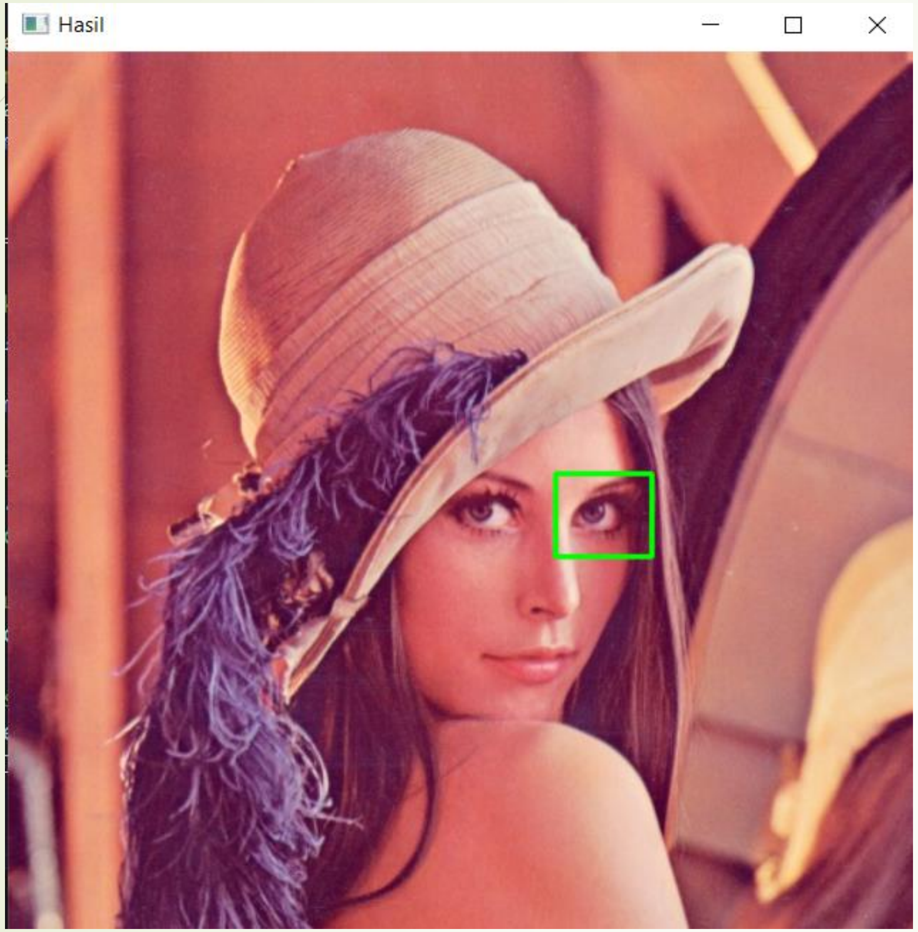
Gambar Utama



Template



Hasil

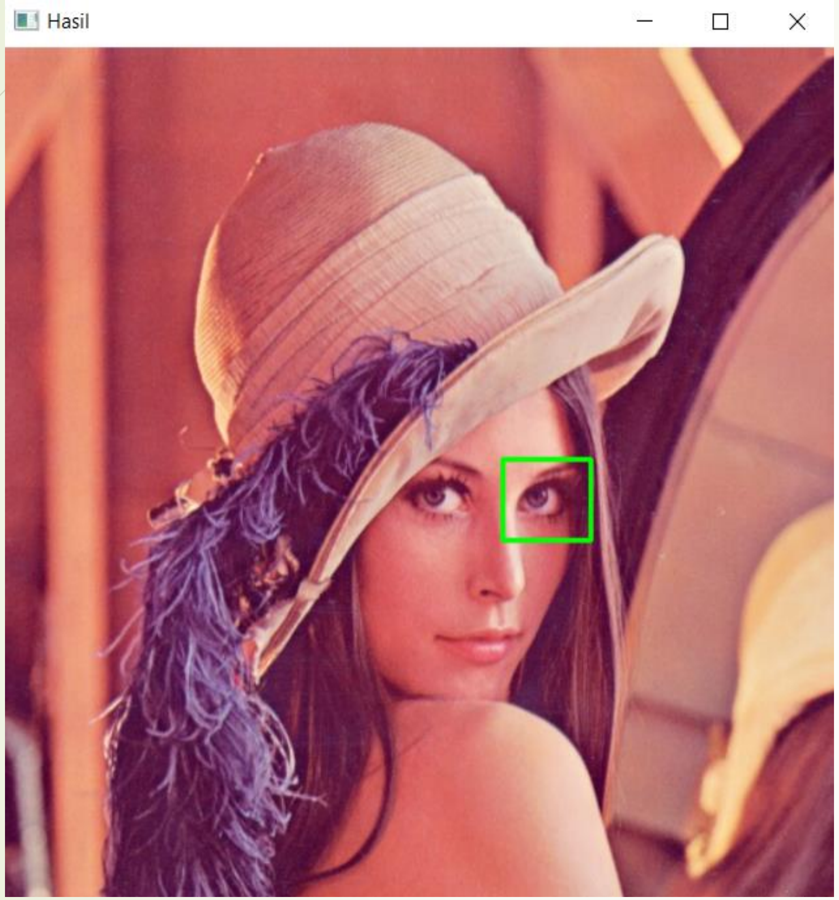


Template Matching Menggunakan SSD

```
import cv2
import numpy as np

def template_matching_ssd(image, template):
    # Konversi gambar ke grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray_template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

    # Ukuran gambar dan template
    height, width = gray_image.shape
    t_height, t_width = gray_template.shape

    # Inisialisasi variabel untuk menyimpan nilai minimum dan koordinat
    min_diff = float('inf')
    min_loc = (0, 0)
    # Loop untuk mencari lokasi dengan perbedaan minimum
    for y in range(height - t_height + 1):
        for x in range(width - t_width + 1):
            # Bagian dari gambar sesuai dengan ukuran template
            roi = gray_image[y:y+t_height, x:x+t_width]

# Menghitung SSD
            ssd = np.sum((roi - gray_template)**2)

            # Memperbarui nilai minimum jika ditemukan perbedaan yang lebih kecil
            if ssd < min_diff:
                min_diff = ssd
                min_loc = (x, y)

    return min_loc

# Baca gambar dan template
image = cv2.imread('gambar_utama.jpg')  # Ganti 'gambar_utama.jpg' dengan nama gambar yang kamu gunakan
template = cv2.imread('template.jpg')   # Ganti 'template.jpg' dengan nama template yang kamu gunakan
```

```
# Panggil fungsi template_matching_ssd
matching_loc = template_matching_ssd(image, template)

# Gambar hasil
result_image = image.copy()
cv2.rectangle(result_image, matching_loc, (matching_loc[0] + template.shape[1], matching_loc[1] + template.shape[0]),
(0, 255, 0), 2)

# Tampilkan gambar hasil
cv2.imshow('Hasil', result_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hasil

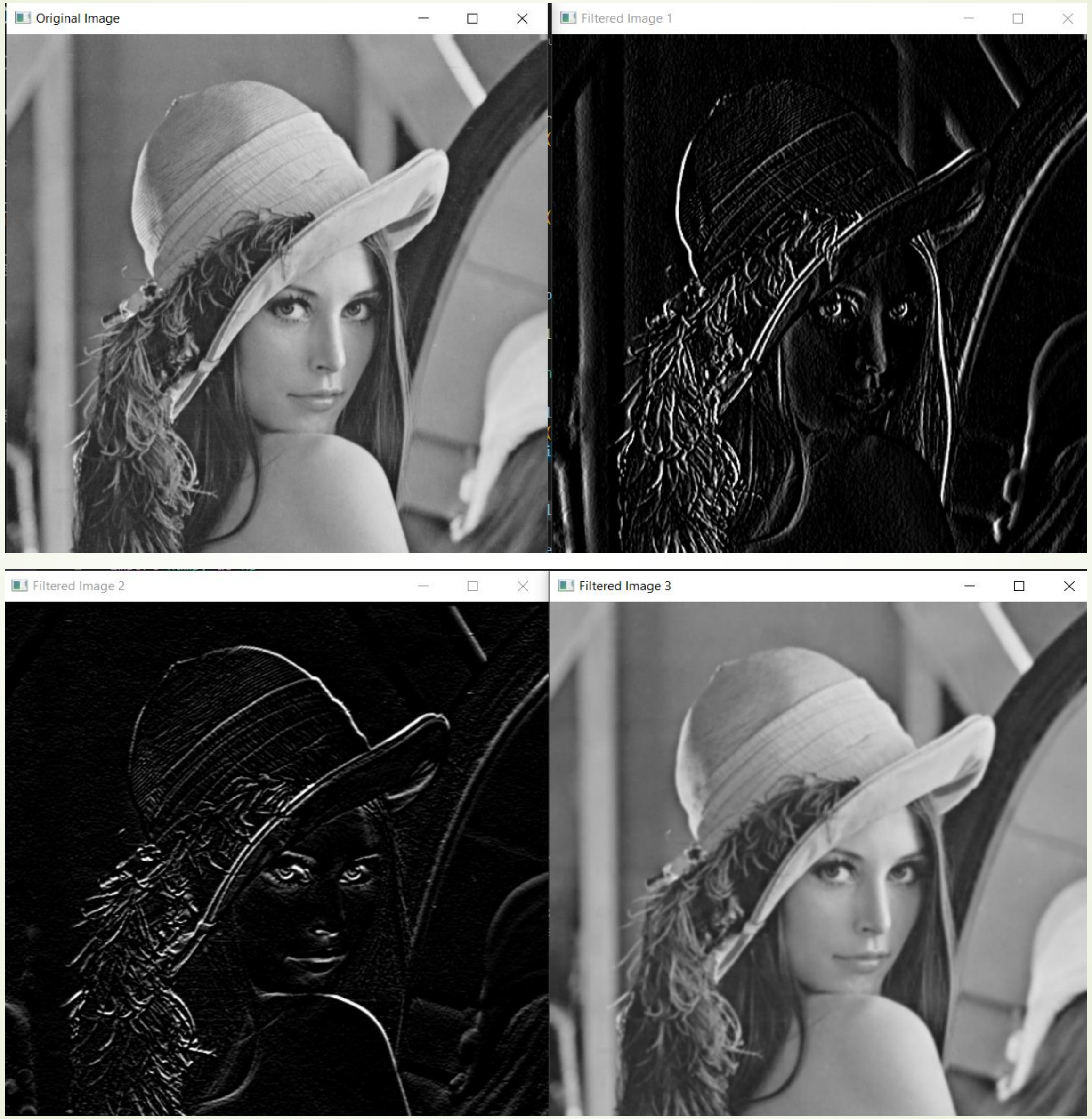


Program Nomor 2 (Filter Bank)

```
import cv2
import numpy as np

def create_sobel_filters():
    sobel_x = np.array([[-1, 0, 1],
                        [-2, 0, 2],
                        [-1, 0, 1]])

    sobel_y = np.array([[-1, -2, -1],
                        [0, 0, 0],
                        [1, 2, 1]])

    return sobel_x, sobel_y

def create_gaussian_filter(size, sigma=1):
    filter_size = size // 2
    gaussian_filter = np.zeros((size, size), dtype=np.float32)
    for x in range(-filter_size, filter_size + 1):
        for y in range(-filter_size, filter_size + 1):
            gaussian_filter[x + filter_size, y + filter_size] = (1 / (2 * np.pi * sigma**2)) * \
                                                        np.exp(-(x**2 + y**2) / (2 * sigma**2))

    return gaussian_filter / np.sum(gaussian_filter)
def apply_filters(image, filters):
    filtered_images = []
    for filter_ in filters:
        filtered = cv2.filter2D(image, -1, filter_)
        filtered_images.append(filtered)
    return filtered_images

# Baca gambar
image = cv2.imread('gambar_utama.jpg')  # Ganti 'contoh_gambar.jpg'
dengan nama gambar yang kamu gunakan

# Konversi gambar ke grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Buat filter bank
sobel_x, sobel_y = create_sobel_filters()
gaussian_filter = create_gaussian_filter(size=5, sigma=1)

# Gabungkan semua filter dalam satu list
filters = [sobel_x, sobel_y, gaussian_filter]

# Terapkan filter bank ke gambar
filtered_images = apply_filters(gray_image, filters)

# Menampilkan hasil
cv2.imshow('Original Image', gray_image)
for i, filtered_image in enumerate(filtered_images):
    cv2.imshow(f'Filtered Image {i+1}', filtered_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

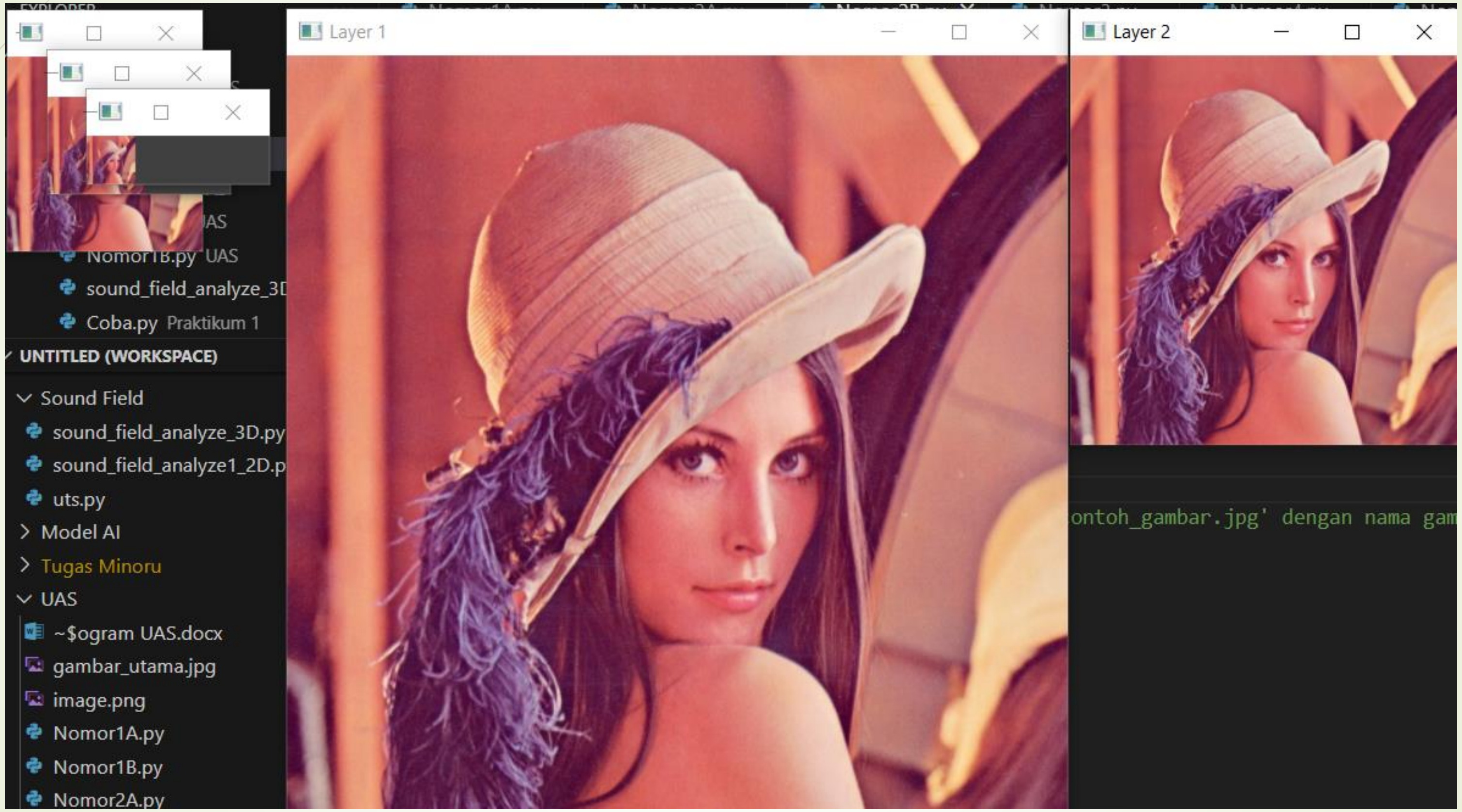
Hasil


Original Image
Filtered Image 1
Filtered Image 2
Filtered Image 3

Program Image Pyramid

```
import cv2


def image_pyramid(image, scale=0.5, min_size=(30, 30)):
    yield image
    while True:
        width = int(image.shape[1] * scale)
        height = int(image.shape[0] * scale)
        image = cv2.resize(image, (width, height))

        if width < min_size[0] or height < min_size[1]:
            break

        yield image

# Baca gambar
image = cv2.imread('gambar_utama.jpg')
# Ganti 'contoh_gambar.jpg' dengan nama gambar yang kamu gunakan
# Buat image pyramid dengan faktor skala 0.5
pyramid = image_pyramid(image)

# Tampilkan gambar-gambar dalam pyramid
i = 1
for resized in pyramid:
    cv2.imshow(f'Layer {i}', resized)
    i += 1

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hasil



```
ontoh_gambar.jpg' dengan nama gam
```

Program Nomor 3 (Image Segmentation menggunakan Metode Ohlander)

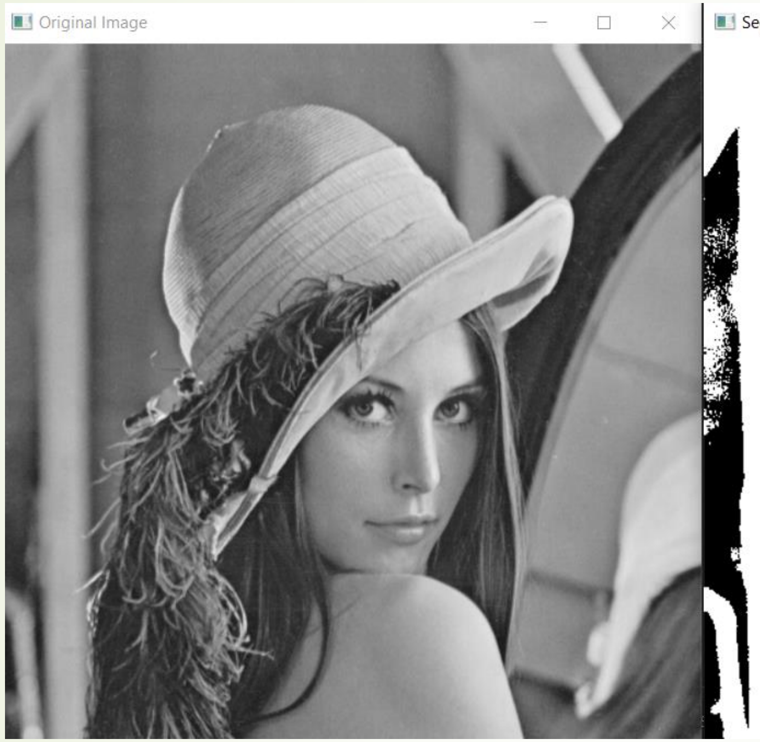
```
import cv2
import numpy as np

def ohlander_segmentation(image, threshold):
    segmented_image = np.zeros(image.shape, dtype=np.uint8)
    labeled_image = np.zeros(image.shape[:2], dtype=np.uint8)

    current_label = 1

    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
            if image[y, x] > threshold:
```

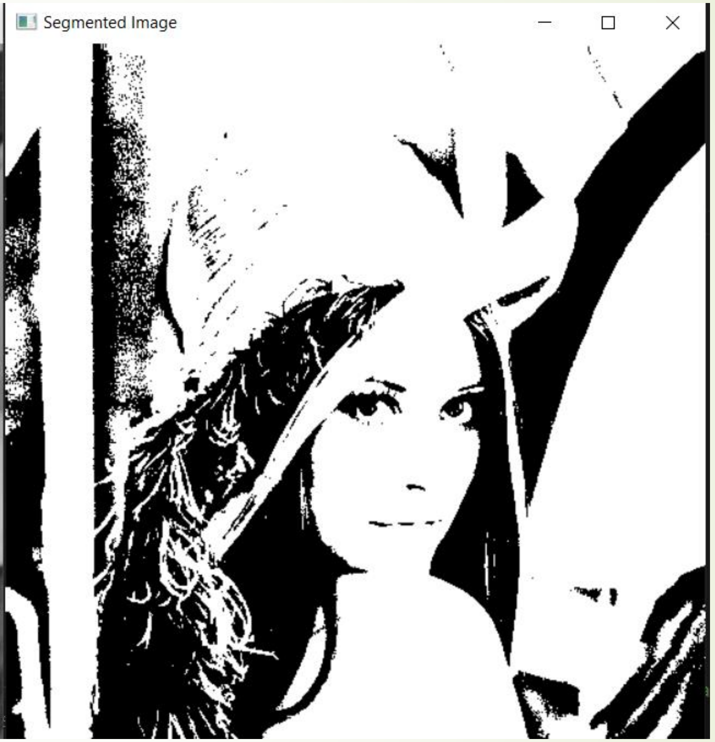
```
if y > 0 and x > 0:
    if labeled_image[y-1, x] != 0:
        labeled_image[y, x] = labeled_image[y-1, x]
    elif labeled_image[y, x-1] != 0:
        labeled_image[y, x] = labeled_image[y, x-1]
    else:
        labeled_image[y, x] = current_label
        current_label += 1
elif y > 0:
    if labeled_image[y-1, x] != 0:
        labeled_image[y, x] = labeled_image[y-1, x]
    else:
        labeled_image[y, x] = current_label
        current_label += 1
elif x > 0:
    if labeled_image[y, x-1] != 0:
        labeled_image[y, x] = labeled_image[y, x-1]
```

```
else:
                       labeled_image[y, x] = current_label
                       current_label += 1
               else:
                   labeled_image[y, x] = current_label
                   current_label += 1

   for y in range(image.shape[0]):
       for x in range(image.shape[1]):
           if labeled_image[y, x] != 0:
               segmented_image[y, x] = 255

   return segmented_image
```

```python
# Baca gambar
image = cv2.imread('gambar_utama.jpg', cv2.IMREAD_GRAYSCALE)  # Ganti 'contoh_gambar.jpg' dengan nama gambar yang kamu gunakan

# Tentukan nilai threshold (contohnya: 100)
threshold_value = 100

# Lakukan segmentasi menggunakan metode Ohlander
segmented_image = ohlander_segmentation(image, threshold_value)

# Tampilkan gambar hasil segmentasi
cv2.imshow('Original Image', image)
cv2.imshow('Segmented Image', segmented_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hasil



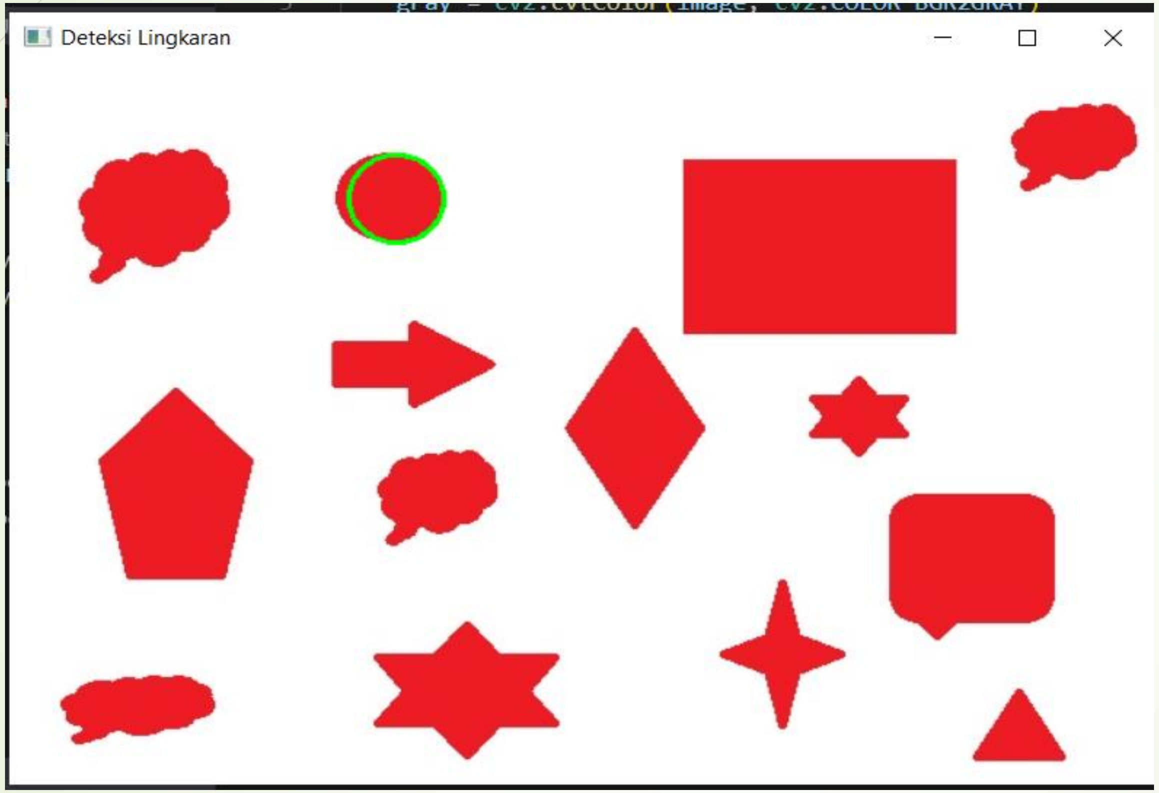


Program Nomor 4 (Mendeteksi Lingkaran)

```
import cv2
import numpy as np

def detect_circles(image, dp=1, minDist=20, param1=50, param2=30, minRadius=0, maxRadius=0):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    circles = cv2.HoughCircles(
        gray, cv2.HOUGH_GRADIENT, dp, minDist, param1=param1, param2=param2, minRadius=minRadius, maxRadius=maxRadius
    )
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for i in circles[0, :]:
            center = (i[0], i[1])
            radius = i[2]
            cv2.circle(image, center, radius, (0, 255, 0), 2)

    return image
```

```
# Baca gambar
image = cv2.imread('shape.jpg')  # Ganti 'contoh_gambar.jpg' dengan nama gambar yang kamu gunakan

# Deteksi lingkaran dengan transformasi Hough
result_image = detect_circles(image)

# Tampilkan gambar hasil
cv2.imshow('Deteksi Lingkaran', result_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hasil



```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Terimakasih