LAPORAN
Visi Komputer dan Pengolahan Citra

Pembahasan Jawaban UAS

Disusun oleh:

Nama : Ahmada Haiz Zakiyil Ilahi
NRP : 1123800010
Kelas : 1 S2 Teknik Elektro
Dosen : Dr Setiawardhana ST, MT

PROGRAM PASCASARJANA TERAPAN

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

PEMBAHASAN JAWABAN UAS

Link Video Demo: https://youtu.be/VvoQ6LKXHD8

1) Template Matching SAD dan SSD

- **Kode Program:**

  Sum Absolute Difference (SAD):

```
import cv2
import numpy as np

def template_matching_sad(image, template):
    # Convert images to grayscale
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    template_gray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

    # Get the dimensions of the images
    image_height, image_width = image_gray.shape
    template_height, template_width = template_gray.shape

    # Initialize variables for best match
    min_sad = float('inf')
    best_match_position = (0, 0)

    # Iterate over possible positions
    for x in range(image_width - template_width + 1):
        for y in range(image_height - template_height + 1):
            # Extract the region from the larger image
            region = image_gray[y:y+template_height,
x:x+template_width]

            # Calculate the Sum of Absolute Differences (SAD)
            sad = np.sum(np.abs(region - template_gray))

            # Update the minimum SAD and best match position
            if sad < min_sad:
                min_sad = sad
                best_match_position = (x, y)

    return best_match_position

if __name__ == "__main__":
    # Read the images
    image = cv2.imread("gambar1.png")
    template = cv2.imread("tmp.png")

    best_match_position = template_matching_sad(image, template)
    print("Best match position:", best_match_position)

    # Get the dimensions of the template
```

```
    template_height, template_width, _ = template.shape

    # Draw a rectangle on the image to highlight the best match
    x, y = best_match_position
    cv2.rectangle(image, (x, y), (x + template_width, y +
template_height), (0, 255, 0), 2)

    # Display the result
    cv2.imshow('Template', template)
    cv2.imshow('Result', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Sum Square Difference (SSD) :

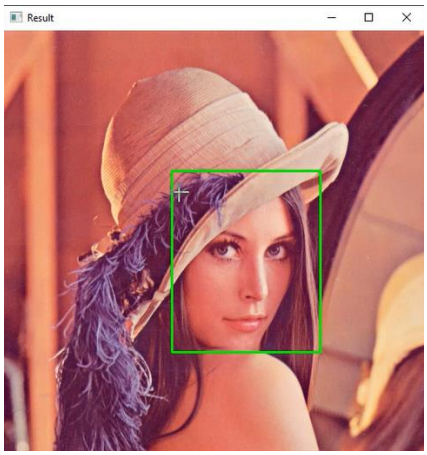
```
import cv2
import numpy as np

def template_matching_ssd(image, template):
    # Convert images to grayscale
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    template_gray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

    # Get the dimensions of the images
    image_height, image_width = image_gray.shape
    template_height, template_width = template_gray.shape

    # Initialize variables for best match
    min_ssd = float('inf')
    best_match_position = (0, 0)

    # Iterate over possible positions
    for x in range(image_width - template_width + 1):
        for y in range(image_height - template_height + 1):
            # Extract the region from the larger image
            region = image_gray[y:y+template_height,
x:x+template_width]

            # Calculate the Sum of Squared Differences (SSD)
            ssd = np.sum((region - template_gray)**2)

            # Update the minimum SAD and best match position
            if ssd < min_ssd:
                min_ssd = ssd
                best_match_position = (x, y)

    return best_match_position

if __name__ == "__main__":
```

```
    # Read the images
    image = cv2.imread("gambar1.png")
    template = cv2.imread("tmp.png")

    best_match_position = template_matching_ssd(image, template)

    print("Best match position:", best_match_position)

    # Get the dimensions of the template
    template_height, template_width, _ = template.shape

    # Draw a rectangle on the image to highlight the best match
    x, y = best_match_position
    cv2.rectangle(image, (x, y), (x + template_width, y +
template_height), (0, 255, 0), 2)

    # Display the result
    cv2.imshow('Template', template)
    cv2.imshow('Result', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

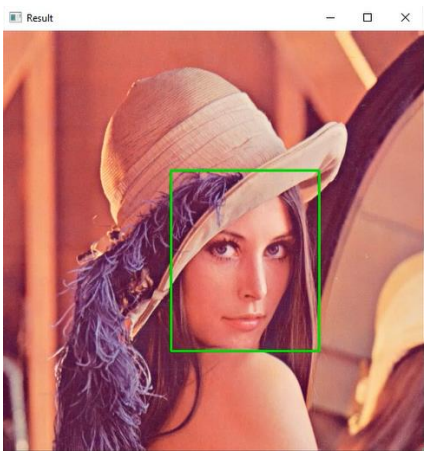
- **Output**:

  Template:

  Hasil:



SAD



SSD

- **Analisa:**

  Kedua program tersebut adalah implementasi dari algoritma template matching, namun menggunakan metode perbandingan berbeda: Program 1 menggunakan Sum of Absolute Differences (SAD), sementara Program 2 menggunakan Sum of Squared Differences (SSD). Pada Program 1, perbandingan dilakukan dengan menghitung total selisih absolut antara intensitas piksel dalam region gambar dan intensitas piksel dalam template. Berikut persamaan SAD:

$$h[m,n] = \sum_{k,l} \left| g[k,l] - f[m+k, n+l] \right|$$

  sedangkan pada Program 2, perbandingan dilakukan dengan menghitung total selisih kuadrat antara intensitas piksel dalam region gambar dan intensitas piksel dalam template. Berikut persamaan SSD:

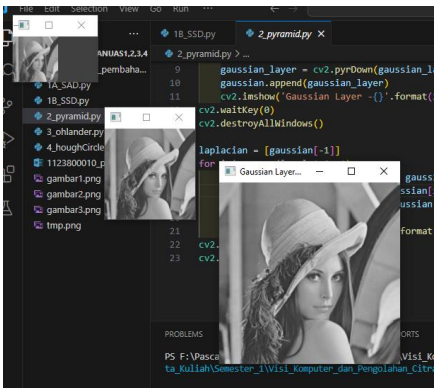
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

2) Image Pyramid

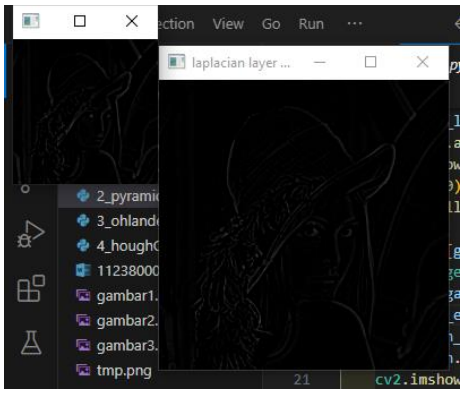
- **Kode Program:**

```
import cv2

image = cv2.imread('gambar1.png',0)
level = 3
gaussian = []

gaussian_layer= image.copy()
for i in range(level):
    gaussian_layer = cv2.pyrDown(gaussian_layer)
    gaussian.append(gaussian_layer)
    cv2.imshow('Gaussian Layer -{}'.format(i),gaussian_layer)
cv2.waitKey(0)
cv2.destroyAllWindows()

laplacian = [gaussian[-1]]
for i in range(level-1,0,-1):
    size = (gaussian[i - 1].shape[1], gaussian[i - 1].shape[0])
    gaussian_expanded = cv2.pyrUp(gaussian[i], dstsize=size)
    laplacian_layer = cv2.subtract(gaussian[i-1], gaussian_expanded)
    laplacian.append(laplacian_layer)
    cv2.imshow('laplacian layer -{}'.format(i-1),laplacian_layer)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Output**:



Gaussian Layer



Laplacian Layer

- **Analisa:**

  Program ini menggunakan piramida Gaussian dan Laplacian untuk menghasilkan gambar berbagai tingkat resolusi dan menunjukkan perbedaan antara tingkat resolusi yang berbeda. Pertama, gambar dibaca dalam skala abu-abu dan disimpan dalam variabel image. Program kemudian membuat piramida Gaussian dengan mengurangkan tingkat resolusi gambar pada setiap iterasi menggunakan cv2.pyrDown, dan menyimpan setiap lapisan Gaussian dalam list gaussian. Lapisan Gaussian ditampilkan menggunakan cv2.imshow. Selanjutnya, program membuat piramida Laplacian dengan menghitung perbedaan antara setiap tingkat resolusi Gaussian dan tingkat resolusi yang diperluas menggunakan cv2.pyrUp. Lapisan Laplacian disimpan dalam list laplacian dan juga ditampilkan menggunakan cv2.imshow. Program memberikan pemahaman visual tentang bagaimana gambar dapat direpresentasikan dalam tingkat resolusi yang berbeda melalui piramida Gaussian dan Laplacian.

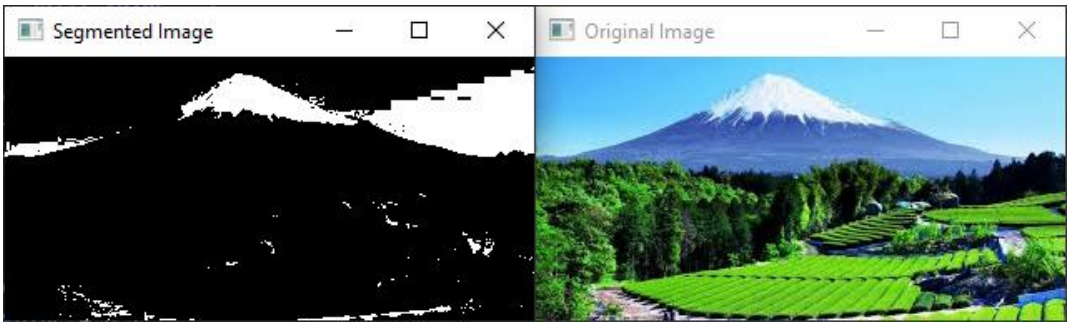
3) Segmentasi gambar menggunakan metode Ohlander's

- **Kode Program:**

```
import cv2

import numpy as np
# Fungsi untuk melakukan segmentasi menggunakan metode Ohlander's
Recursive Histogram-Based Clustering
def ohlander_clustering(image, threshold):
    if len(image.shape) > 2:
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray_image = image.copy()
    # Mendapatkan histogram dari gambar
    hist = cv2.calcHist([gray_image], [0], None, [256], [0, 256])
```

```
    # Mencari nilai untuk clustering
    split_value = 0
    max_val = np.max(hist)
    for i in range(255, 0, -1):
        if hist[i] > threshold * max_val:
            split_value = i
            break

    # Segmentasi gambar
    segmented_image = np.zeros_like(gray_image)
    segmented_image[gray_image >= split_value] = 255

    return segmented_image

input_image = cv2.imread('gambar2.png')
threshold_value = 0.7

segmented_image = ohlander_clustering(input_image, threshold_value)
cv2.imshow('Original Image', input_image)
cv2.imshow('Segmented Image', segmented_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Output**:



Gambar hasil segmentasi gambar menggunakan metode Ohlander's

- **Analisa:**

Program tersebut bertujuan untuk melakukan segmentasi pada gambar menggunakan metode Ohlander's Recursive Histogram-Based Clustering. Segmentasi adalah proses membagi citra menjadi beberapa bagian (segment) berdasarkan karakteristik tertentu. Metode yang digunakan dalam program ini, yaitu Ohlander's Recursive Histogram-Based Clustering, berfokus pada pemisahan piksel-piksel dalam gambar berdasarkan histogram intensitasnya. Pertama, gambar dibaca dan, jika perlu, dikonversi menjadi citra skala abu-abu. Kemudian, histogram intensitas gambar dihitung. Selanjutnya, program mencari nilai ambang yang digunakan untuk melakukan clustering. Proses

clustering dilakukan dengan memisahkan piksel-piksel pada gambar berdasarkan nilai ambang tersebut. Piksel-piksel dengan intensitas di atas nilai ambang diberi nilai putih (255), sedangkan yang di bawah nilai ambang diberi nilai hitam (0).

4) Deteksi Lingkaran menggunakan Transformasi Hough

- **Kode Program:**

```
import numpy as np

import cv2

# Read image
img = cv2.imread('gambar3.png')
# Smooth it
img = cv2.medianBlur(img,3)
img_copy = img.copy()
# Convert to greyscale
img_gray = cv2.cvtColor(img_copy,cv2.COLOR_BGR2GRAY)
# Apply Hough transform to greyscale image
circles = cv2.HoughCircles(img_gray,cv2.HOUGH_GRADIENT,1,20,
                    param1=60,param2=40,minRadius=0,maxRadius=0)
circles = np.uint16(np.around(circles))
# Draw the circles
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(img,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(img,(i[0],i[1]),2,(0,0,255),3)
cv2.imshow('detected circles',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Output**:

Gambar Hasil Deteksi Lingkaran

- **Analisa:**

  Program tersebut merupakan implementasi deteksi lingkaran (circular object detection) menggunakan transformasi Hough pada citra. Kode ini dimulai dengan membaca gambar yang diberi nama 'gambar3.png' dan menerapkan operasi median blur untuk menghaluskan gambar tersebut. Selanjutnya, gambar diubah menjadi grayscale, dan dilakukan transformasi Lingkaran Hough untuk mendeteksi lingkaran. Parameter dari fungsi HoughCircles mengontrol sensitivitas deteksi lingkaran. Lingkaran yang terdeteksi kemudian digambar pada gambar asli, dengan lingkaran luar berwarna hijau dan pusatnya berwarna merah. Gambar hasilnya ditampilkan menggunakan fungsi imshow dari OpenCV, dan program menunggu penekanan tombol sebelum menutup jendela tampilan. Kode ini memperlihatkan penggunaan teknik visi komputer untuk deteksi lingkaran dalam sebuah gambar, yang dapat bermanfaat dalam berbagai aplikasi seperti pengenalan objek atau analisis gambar.