Nama: Andi Cleopatra Maryam Jamila

Nim: 1103213071

Tugas Analisis Week 11

1. Analisis Python

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
# Fungsi untuk menampilkan gambar
def show_image(title, img, cmap=None):
    plt.figure(figsize=(8, 6))
    plt.imshow(img, cmap=cmap)
    plt.title(title)
    plt.axis('off')
    plt.show()
```

```
# Load gambar
image_path = "/content/sample_data/pelangii.jpg"
img = cv2.imread(image_path)
if img is None:
    raise FileNotFoundError(f"Gambar di {image_path} tidak ditemukan!")

# Ubah warna gambar dari BGR ke RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
# === 1. Ekstraksi garis dengan Hough Transform ===
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)
lines = cv2.HoughLines(edges, 1, np.pi / 180, 200)
line_img = img_rgb.copy()
if lines is not None:
    for rho, theta in lines[:, 0]:
        a, b = np.cos(theta), np.sin(theta)
        x0, y0 = a * rho, b * rho
        x1, y1 = int(x0 + 1000 * (-b)), int(y0 + 1000 * a)
        x2, y2 = int(x0 - 1000 * (-b)), int(y0 - 1000 * a)
        cv2.line(line_img, (x1, y1), (x2, y2), (255, 0, 0), 2)
show_image("Hough Transform - Line Detection", line_img)
```

Output:

Analisis:
menggunakan OpenCV untuk mendeteksi garis pada gambar melalui algoritma Hough Transform. Setelah gambar diubah ke grayscale dan tepiannya diekstraksi dengan deteksi tepi Canny, algoritma Hough Transform diterapkan untuk mengidentifikasi garis berdasarkan representasi parameterik (rho, theta). Garis-garis yang terdeteksi kemudian digambar pada salinan gambar asli dengan warna merah, dan hasilnya divisualisasikan menggunakan Matplotlib. Proses ini sangat efektif untuk mendeteksi garis lurus pada gambar, seperti tepi jalan atau struktur bangunan, yang dapat digunakan dalam aplikasi seperti navigasi robot atau analisis citra.

```
# === 2. Template Matching untuk Deteksi Objek ===
template_path = "/content/sample_data/pelangii.jpg"
template = cv2.imread(template_path, 0)
if template is None:
    raise FileNotFoundError(f"Template di {template_path} tidak ditemukan!")
w, h = template.shape[::-1]
res = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
threshold = 0.8
loc = np.where(res >= threshold)
template_img = img_rgb.copy()
for pt in zip(*loc[::-1]):
    cv2.rectangle(template_img, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 2)
show_image("Template Matching", template_img)
```

Output:

Analisis:
Menggunakan metode Template Matching dari OpenCV untuk mendeteksi keberadaan objek tertentu dalam gambar berdasarkan pola (template) yang diberikan. Template diubah menjadi grayscale, dan fungsi cv2.matchTemplate menghitung kesesuaian antara template dan area gambar menggunakan metode normalisasi koefisien korelasi (TM_CCOEFF_NORMED). Lokasi dengan nilai kesesuaian di atas ambang batas (threshold = 0.8) dianggap sebagai kecocokan, dan persegi panjang hijau digambar pada lokasi-lokasi tersebut pada gambar asli. Hasil akhirnya divisualisasikan menggunakan Matplotlib, menjadikan metode ini efektif untuk mendeteksi objek dengan bentuk yang sama persis dengan template, meskipun rentan terhadap perubahan skala, rotasi, atau pencahayaan.

```python
# === 3. Pembuatan Pyramid Gambar ===
lower_reso = cv2.pyrDown(img_rgb)
higher_reso = cv2.pyrUp(img_rgb)
show_image("Lower Resolution (PyrDown)", lower_reso)
show_image("Higher Resolution (PyrUp)", higher_reso)
```

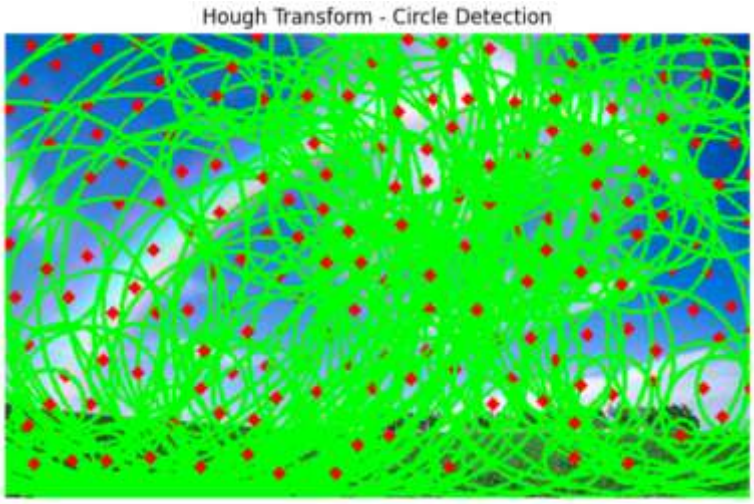
Output:

Analisis:

Menggunakan metode piramida gambar OpenCV untuk mengubah resolusi gambar. Fungsi cv2.pyrDown menurunkan resolusi gambar dengan mengurangi jumlah piksel melalui penghalusan dan subsampling, menghasilkan gambar beresolusi lebih rendah yang dapat digunakan untuk pemrosesan lebih cepat. Sebaliknya, cv2.pyrUp meningkatkan resolusi gambar dengan interpolasi, yang dapat digunakan untuk rekonstruksi atau analisis multi-skala. Hasil dari kedua transformasi ini ditampilkan menggunakan Matplotlib. Proses ini sering digunakan dalam aplikasi seperti deteksi objek multi-skala atau kompresi gambar.

```python
# === 4. Deteksi Lingkaran dengan Hough Transform ===
circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 20, param1=50, param2=30, minRadius=0, maxRadius=0)
circle_img = img_rgb.copy()
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        cv2.circle(circle_img, (i[0], i[1]), i[2], (0, 255, 0), 2)
        cv2.circle(circle_img, (i[0], i[1]), 2, (255, 0, 0), 3)
show_image("Hough Transform - Circle Detection", circle_img)
```

Output:



Analisis:
Menggunakan metode Hough Transform untuk mendeteksi lingkaran dalam citra. Fungsi cv2.HoughCircles digunakan untuk mendeteksi lingkaran pada gambar dalam ruang abu-abu (gray). Parameter seperti param1, param2, dan jangkauan radius (minRadius dan maxRadius) diatur untuk mengoptimalkan deteksi. Jika lingkaran ditemukan, fungsi ini menggambar lingkaran pada gambar asli menggunakan cv2.circle, dengan warna hijau untuk lingkaran luar dan biru untuk pusatnya. Hasil akhir ditampilkan dengan menggunakan fungsi show_image. Teknik ini efektif untuk mendeteksi objek berbentuk lingkaran dalam citra, seperti peluru, roda, atau tutup botol, dalam berbagai kondisi gambar.

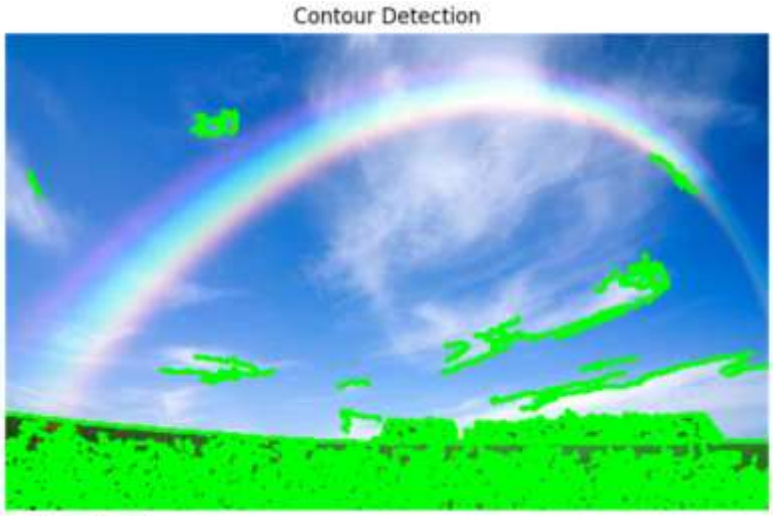
```
# === 5. Ekstraksi Warna Dominan pada Gambar ===
def extract_dominant_color(image, k=3):
    pixels = image.reshape((-1, 3))
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(pixels)
    dominant_color = kmeans.cluster_centers_.astype(int)
    return dominant_color

colors = extract_dominant_color(img_rgb, k=5)
print("Dominant Colors (RGB):")
for i, color in enumerate(colors):
    print(f"Color {i + 1}: {color}")
```

```
# === 6. Deteksi Kontur pada Gambar ===
contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contour_img = img_rgb.copy()
cv2.drawContours(contour_img, contours, -1, (0, 255, 0), 2)
show_image("Contour Detection", contour_img)

print("Simulasi selesai!")
```

Output:




```
Simulasi selesai!
```

Analisis:
Menggabungkan dua teknik penting dalam pengolahan citra: ekstraksi warna dominan dan deteksi kontur. Fungsi extract_dominant_color menggunakan algoritma KMeans untuk mengelompokkan piksel citra menjadi beberapa klaster dan menemukan warna dominan berdasarkan nilai RGB. Dengan parameter k=5, fungsi ini mengidentifikasi lima warna utama dalam citra. Selanjutnya, deteksi kontur dilakukan menggunakan cv2.findContours untuk menemukan kontur objek dalam citra yang sudah diproses dengan deteksi tepi. Kontur tersebut digambar pada gambar asli dengan warna hijau, memberikan gambaran visual tentang bentuk dan batas objek dalam gambar. Kedua teknik ini berguna dalam analisis citra untuk mengidentifikasi objek serta fitur visual utama seperti warna dan bentuk.

2. Analisis Webots
   Analisis:
   Simulasi Webots yang berfokus pada ekstraksi data LiDAR dan deteksi hambatan bertujuan untuk memodelkan cara robot memahami dan merespons lingkungan sekitarnya menggunakan sensor LiDAR (Light Detection and Ranging). LiDAR bekerja dengan mengirimkan sinar laser untuk mengukur jarak ke objek di sekitar robot, menghasilkan peta 3D atau representasi jarak yang sangat akurat. Data yang diperoleh dari LiDAR kemudian digunakan untuk mendeteksi hambatan atau objek di jalur robot. Dalam simulasi ini, algoritma deteksi hambatan akan memproses data LiDAR untuk mengidentifikasi objek yang berada dalam jarak tertentu, memungkinkan robot untuk mengambil keputusan seperti menghindari tabrakan atau merencanakan jalur yang optimal. Teknik ini sangat penting dalam pengembangan robot otonom, karena memungkinkan navigasi yang aman dan efisien dalam lingkungan yang dinamis.