ANANLISIS TUGAS 1 DAN 2

1. Ekstraksi Garis dengan Hough Transform

Tujuan:
Mendeteksi dan mengekstrak garis lurus pada gambar menggunakan metode Hough Line Transform.

Proses:
- Menggunakan metode Canny Edge Detection untuk menemukan tepi gambar.
- Garis lurus diekstrak berdasarkan transformasi Hough.

Kode:

```
def hough_line_transform(image_path):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150, apertureSize=3)

    lines = cv2.HoughLines(edges, 1, np.pi / 180, 200)
    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(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Hough Line Transform')
    plt.axis('off')
    plt.show()
```

2. Template Matching untuk Deteksi Objek

Tujuan:
Mendeteksi objek tertentu dalam gambar menggunakan template matching.

Proses:

- Template dicari di dalam gambar utama menggunakan metode Normalized Cross-Correlation.

Kode:

```
def template_matching(image_path, template_path):
    img = cv2.imread(image_path, 0)
    template = cv2.imread(template_path, 0)
    w, h = template.shape[::-1]

    result = cv2.matchTemplate(img, template, cv2.TM_CCOEFF_NORMED)
    _, _, _, max_loc = cv2.minMaxLoc(result)
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)

    img_color = cv2.imread(image_path)
    cv2.rectangle(img_color, top_left, bottom_right, (0, 0, 255), 2)

    plt.imshow(cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB))
    plt.title('Template Matching')
    plt.axis('off')
    plt.show()
```

3. Pembuatan Pyramid Gambar

Tujuan:
Menghasilkan gambar dengan resolusi lebih rendah menggunakan Image Pyramid.

Proses:
- Menggunakan Gaussian Pyramid untuk downscaling gambar secara bertahap.

Kode:

```
def image_pyramid(image_path):
    img = cv2.imread(image_path)
    plt.figure(figsize=(10, 5))

    for i in range(3):
        plt.subplot(1, 3, i+1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title(f'Level {i}')
        plt.axis('off')
```

```
        img = cv2.pyrDown(img)

    plt.show()
```

4. Deteksi Lingkaran Menggunakan Hough Transform

Tujuan:
Mendeteksi lingkaran dalam gambar menggunakan Hough Circle Transform.

Proses:
- Menggunakan metode Hough Gradient untuk menemukan parameter lingkaran.

Kode:

```
def hough_circle_transform(image_path):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
                               param1=100, param2=30, minRadius=20, maxRadius=100)
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for i in circles[0, :]:
            cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 2)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Hough Circle Transform')
    plt.axis('off')
    plt.show()
```

5. Ekstraksi Warna Dominan pada Gambar

Tujuan:
Menentukan warna dominan dalam gambar menggunakan metode K-Means Clustering.

Proses:
- Piksel gambar dikelompokkan menjadi beberapa kluster warna.

Kode:

```
def dominant_color_extraction(image_path, k=3):
```

```
    img = cv2.imread(image_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_flat = img.reshape((-1, 3))

    kmeans = KMeans(n_clusters=k)
    kmeans.fit(img_flat)

    colors = kmeans.cluster_centers_.astype(int)
    plt.imshow([colors / 255.0])
    plt.title('Dominant Colors')
    plt.axis('off')
    plt.show()
```

6. Deteksi Kontur pada Gambar

Tujuan:
Mendeteksi kontur objek berdasarkan tepi gambar.

Proses:
- Menggunakan Thresholding untuk binerisasi gambar.
- Kontur digambar menggunakan metode findContours.

Kode:

```
def contour_detection(image_path):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(img, contours, -1, (0, 255, 0), 2)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Contour Detection')
    plt.axis('off')
    plt.show()
```

Analisis Simulasi Webots: Lidar Data Extraction and Obstacle Detection

1. Analisis Umum

Simulasi Lidar Data Extraction and Obstacle Detection dalam Webots berfokus pada pemanfaatan sensor LIDAR dan sensor ultrasonik untuk mendeteksi rintangan serta menghindarinya secara dinamis. Robot dalam simulasi ini dilengkapi dengan sensor LIDAR yang digunakan untuk mengekstraksi data jarak di sekitar lingkungan robot, sedangkan sensor ultrasonik digunakan untuk pengendalian kecepatan berdasarkan jarak kiri dan kanan. Kode Python yang digunakan mencakup inisialisasi perangkat, pembacaan data sensor, serta pengaturan kecepatan roda untuk melakukan manuver penghindaran rintangan.

2. Penjelasan Rinci Kode Python

a. Inisialisasi Perangkat

- **Robot dan Sensor LIDAR**:
  Sensor LIDAR diinisialisasi dan diaktifkan dengan `enable()` serta mode point cloud diaktifkan untuk mendapatkan data jarak dari lingkungan sekitar.

```
robot = Robot()
lidar = robot.getDevice("lidar")
lidar.enable(TIME_STEP)
lidar.enablePointCloud()
```

- **Sensor Ultrasonik**:
  Dua sensor jarak (kiri dan kanan) diaktifkan untuk memonitor jarak rintangan dan memberikan masukan ke pengontrol kecepatan.

```
us = [robot.getDevice("us0"), robot.getDevice("us1")]
for sensor in us:
  sensor.enable(TIME_STEP)
```

- **Motor Roda**:
  Motor roda kiri dan kanan diatur dalam mode kecepatan dengan menggunakan

`setPosition(float('inf'))`. Kecepatan awal motor disetel ke 0.

```
left_motor = robot.getDevice("left wheel motor")
right_motor = robot.getDevice("right wheel motor")
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))
left_motor.setVelocity(0.0)
right_motor.setVelocity(0.0)
```

b. Fungsi Ekstraksi Data

- **Pembacaan Data LIDAR**:
  Fungsi ini membaca data jarak dari LIDAR dan menampilkan 10 nilai pertama sebagai representasi jarak objek di sekeliling robot.

```
def extract_lidar_data():
    lidar_data = lidar.getRangeImage()
    print(f"Lidar Data: {lidar_data[:10]}...")
    return lidar_data
```

- **Pembacaan Data Sensor Jarak**:
  Nilai jarak dari sensor kiri dan kanan dibaca dan digunakan untuk pengambilan keputusan dalam pengaturan kecepatan roda.

```
def read_distance_sensors():
    distances = [sensor.getValue() for sensor in us]
    print(f"Distance Sensor Readings Left={distances[LEFT]:.2f},
Right={distances[RIGHT]:.2f}")
    return distances
```

c. Penghitungan Kecepatan Dinamis

Kecepatan roda kiri dan kanan dihitung berdasarkan jarak yang terdeteksi oleh sensor ultrasonik dengan koefisien empiris.

```
def compute_speeds(us_values):
    speed = [0.0, 0.0]
    for i in range(2):
        for k in range(2):
            speed[i] += us_values[k] * coefficients[i][k]
    return speed
```

d. Loop Utama

Dalam loop simulasi, robot membaca data LIDAR dan sensor jarak, kemudian menghitung kecepatan roda untuk melakukan penghindaran rintangan.

```
while robot.step(TIME_STEP) != -1:
    lidar_data = extract_lidar_data()
    us_values = read_distance_sensors()
    speeds = compute_speeds(us_values)
    left_motor.setVelocity(base_speed + speeds[LEFT])
    right_motor.setVelocity(base_speed + speeds[RIGHT])
```

3. Kesimpulan

Kode ini mengimplementasikan pendekatan reaktif sederhana untuk penghindaran rintangan. Sensor ultrasonik memainkan peran utama dalam mengatur kecepatan berdasarkan jarak rintangan di kiri dan kanan, sedangkan data LIDAR memberikan wawasan lebih lanjut tentang jarak di sekitar robot. Dengan loop utama yang terus memantau lingkungan, robot dapat melakukan pergerakan dinamis untuk menavigasi area secara mandiri. Simulasi ini menunjukkan bagaimana kombinasi LIDAR dan ultrasonik dapat digunakan secara efektif untuk pengambilan keputusan dalam pengendalian robot berbasis jarak.