1. Deteksi Blob Warna (Merah, Hijau, dan Biru)

Pada simulasi ini, kita menggunakan pemrosesan citra untuk mendeteksi blob warna tertentu (merah, hijau, biru) menggunakan teknik thresholding pada ruang warna HSV (Hue, Saturation, Value), karena HSV lebih robust dalam mendeteksi warna dibandingkan dengan ruang warna RGB.

Kode Lengkap:

```
import cv2
import numpy as np

# Inisialisasi kamera (misalnya, webcam atau kamera robot)
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Convert to HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Threshold untuk warna merah
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red, upper_red)

    # Threshold untuk warna hijau
    lower_green = np.array([35, 100, 100])
    upper_green = np.array([85, 255, 255])
    mask_green = cv2.inRange(hsv, lower_green, upper_green)

    # Threshold untuk warna biru
    lower_blue = np.array([100, 150, 0])
    upper_blue = np.array([140, 255, 255])
    mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

    # Gabungkan mask untuk ketiga warna
    mask_combined = mask_red | mask_green | mask_blue

    # Terapkan mask ke gambar asli
    result = cv2.bitwise_and(frame, frame, mask=mask_combined)

    # Tampilkan hasil
    cv2.imshow('Detected Blobs', result)
```

```
    # Keluar jika tekan 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Penjelasan:

- **Proses Konversi ke HSV**: Gambar yang diambil dari kamera pertama kali dikonversi ke ruang warna HSV untuk memudahkan deteksi warna. Dalam HSV, warna dikelompokkan berdasarkan Hue (warna), Saturation (kecerahan warna), dan Value (intensitas).

- **Thresholding untuk Warna**:

    - Setiap warna (merah, hijau, biru) memiliki rentang nilai HSV yang berbeda. `cv2.inRange()` digunakan untuk menghasilkan mask yang memisahkan area yang berisi warna yang ditargetkan.
- **Gabungkan Mask**: Mask untuk merah, hijau, dan biru digabungkan untuk mendeteksi ketiga warna sekaligus dalam gambar.

- **Hasil**: Gambar yang berisi hanya area dengan warna merah, hijau, atau biru akan ditampilkan. Ini adalah contoh dasar dari deteksi blob warna.

Analisis:

- **Kelebihan**: Penggunaan HSV lebih tahan terhadap variasi pencahayaan dibandingkan RGB.
- **Keterbatasan**: Deteksi dapat terganggu oleh perubahan kondisi pencahayaan atau adanya objek dengan warna yang mirip

2. Fokus Kamera Berdasarkan Objek yang Ada di Depannya

Simulasi ini melibatkan penyesuaian fokus kamera berdasarkan jarak objek di depannya, yang biasanya dilakukan dengan menggunakan sensor kedalaman atau depth map.

Kode Lengkap:

```
import cv2
import numpy as np

# Kamera RGB dan sensor kedalaman
cap_rgb = cv2.VideoCapture(0)
cap_depth = cv2.VideoCapture(1)  # Asumsi sensor kedalaman ada pada device lain

while True:
```

```
    ret_rgb, frame_rgb = cap_rgb.read()
    ret_depth, frame_depth = cap_depth.read()

    if not ret_rgb or not ret_depth:
        break

    # Misalkan frame_depth adalah citra grayscale yang mewakili kedalaman
    # Temukan titik dengan kedalaman terdekat
    min_depth = np.min(frame_depth)
    focus_distance = min_depth  # Sesuaikan berdasarkan kedalaman objek

    # Terapkan filter atau kontrol fokus berdasarkan focus_distance
    print(f'Focusing on distance: {focus_distance} units')

    # Tampilkan hasil
    cv2.imshow('RGB Camera', frame_rgb)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap_rgb.release()
cap_depth.release()
cv2.destroyAllWindows()
```

Penjelasan:

- **Sensor Kedalaman**: Untuk simulasi ini, robot memiliki dua kamera: satu untuk gambar RGB dan satu untuk sensor kedalaman (misalnya, menggunakan depth sensor seperti Kinect atau kamera stereo).

- **Pengukuran Kedalaman**: Kedalaman dihitung berdasarkan citra dari sensor kedalaman. Kedalaman minimum objek yang terdeteksi di layar digunakan untuk menentukan jarak fokus kamera.

- **Penyesuaian Fokus**: Di dunia nyata, penyesuaian fokus kamera dilakukan dengan mekanisme optik berdasarkan jarak objek, tetapi di sini kita hanya mensimulasikan penyesuaian tersebut dengan mengukur kedalaman dan menggunakan nilai kedalaman untuk memodulasi parameter fokus (secara teoritis).

Analisis:

- **Kelebihan**: Penyesuaian fokus berdasarkan kedalaman membuat robot lebih efisien dalam menangkap objek di jarak yang tepat.
- **Keterbatasan**: Membutuhkan perangkat keras tambahan (seperti sensor kedalaman), dan proses ini lebih kompleks jika digunakan dalam situasi dunia nyata.

3. Deteksi Blob Berwarna dengan Efek Motion Blur

Efek motion blur mensimulasikan gangguan visual yang terjadi ketika objek atau kamera bergerak cepat. Motion blur dapat ditambahkan untuk mensimulasikan kondisi tertentu dalam pemrosesan citra.

Kode Lengkap:

```
import cv2
import numpy as np

def motion_blur(image, degree=15):
    kernel = np.zeros((degree, degree))
    kernel[int((degree - 1)/2), :] = np.ones(degree)
    kernel /= degree
    return cv2.filter2D(image, -1, kernel)

# Deteksi blob dan motion blur
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Deteksi warna (misal: merah)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(frame, frame, mask=mask_red)

    # Terapkan motion blur
    blurred_result = motion_blur(result)

    # Tampilkan hasil
    cv2.imshow('Blob with Motion Blur', blurred_result)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Penjelasan:

- **Motion Blur**: Fungsi `motion_blur()` membuat kernel blur linier, yang kemudian diterapkan pada gambar untuk memberi efek blur yang terlihat seperti objek bergerak.

- **Deteksi Warna**: Kami mendeteksi warna merah dengan cara yang sama seperti di simulasi pertama, tetapi kali ini gambar hasil deteksi warna dikenakan efek motion blur.

- **Efek Blur**: Motion blur memberikan tampilan visual seolah-olah kamera atau objek bergerak cepat, yang bisa digunakan untuk mensimulasikan kondisi nyata.

Analisis:

- **Kelebihan**: Efek motion blur dapat membuat simulasi lebih realistis.
- **Keterbatasan**: Pengaruh motion blur pada deteksi objek bisa mengurangi keakuratan deteksi, karena citra menjadi kabur.

4. Deteksi Blob Berwarna dengan Noise Mask

Pada simulasi ini, noise ditambahkan untuk mensimulasikan gangguan dalam citra. Noise ini bisa berupa gangguan acak yang mempengaruhi kualitas gambar.

Kode Lengkap:

```
import cv2
import numpy as np

def add_noise(image, noise_factor=0.2):
    noise = np.random.randn(*image.shape) * noise_factor
    noisy_image = np.clip(image + noise, 0, 255).astype(np.uint8)
    return noisy_image

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Deteksi warna
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(frame, frame, mask=mask_red)
```

```
    # Tambahkan noise
    noisy_result = add_noise(result)

    # Tampilkan hasil
    cv2.imshow('Blob with Noise Mask', noisy_result)

    if


cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release() cv2.destroyAllWindows()
```

Penjelasan:
- **Noise**: Fungsi `add_noise()` menghasilkan noise acak berdasarkan distribusi normal, yang kemudian ditambahkan pada gambar yang sudah terdeteksi blob warnanya.

- **Gangguan pada Citra**: Noise ini mensimulasikan gangguan yang terjadi di dunia nyata, misalnya gangguan sinyal atau kualitas gambar yang buruk.

Analisis:
- **Kelebihan**: Membantu mempersiapkan model deteksi untuk menghadapi kondisi dunia nyata yang tidak sempurna.
- **Keterbatasan**: Noise dapat merusak akurasi deteksi jika tidak ditangani dengan baik.

5. Deteksi Objek dengan Kamera dan Pengenalan Objek
Simulasi ini menggunakan metode pembelajaran mesin untuk melakukan pengenalan objek, seperti YOLO (You Only Look Once), yang dapat mendeteksi berbagai objek dalam citra secara real-time.

Kode Lengkap:

```
import cv2
import numpy as np

# Load pre-trained YOLO model
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
```

```
    # Prepare image for YOLO
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Process detections
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                # Deteksi objek
                # (bounding box, etc.)
                pass

    # Tampilkan hasil
    cv2.imshow('Object Detection', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Penjelasan:

- **YOLO**: YOLO adalah metode deteksi objek real-time yang dapat mendeteksi banyak objek dalam satu frame.
- **Model Pre-trained**: Kami memanfaatkan model YOLO yang sudah dilatih sebelumnya (`yolov3.weights` dan `yolov3.cfg`) untuk mendeteksi objek.
- **Proses Deteksi**: Gambar yang diambil dari kamera diproses untuk mendeteksi objek. Jika confidence deteksi lebih besar dari threshold (misalnya, 50%), objek akan dikenali.

Analisis:

- **Kelebihan**: YOLO sangat cepat dan efisien dalam mendeteksi berbagai objek secara real-time.
- **Keterbatasan**: Memerlukan perangkat keras yang cukup kuat untuk menjalankan model pre-trained dengan baik.

6. Implementasi Segmentasi Kamera pada Robot menggunakan Webots

Implementasi ini melibatkan segmentasi citra pada robot yang digunakan dalam simulasi Webots. Di sini, segmentasi citra digunakan untuk memisahkan objek dari latar belakang.

Kode Segmentasi menggunakan Webots (Python API):

```
from controller import Robot, Camera

# Inisialisasi robot dan kamera
robot = Robot()
camera = robot.getCamera("camera")
camera.enable(32)

# Loop utama
while robot.step(32) != -1:
    image = camera.getImage()

    # Segmentasi citra menggunakan thresholding
    # Misalnya, thresholding berdasarkan intensitas warna
    width = camera.getWidth()
    height = camera.getHeight()

    for x in range(width):
        for y in range(height):
            pixel = camera.getImagePixel(image, x, y)
            r, g, b = pixel[0], pixel[1], pixel[2]
            if r > 100 and g < 50 and b < 50:
                # Deteksi pixel merah
                pass
    # Proses lebih lanjut atau kirim data ke controller lain
```

Penjelasan:

- **Segmentasi**: Di sini, segmentasi dilakukan dengan menganalisis nilai intensitas piksel untuk menentukan apakah mereka termasuk dalam kategori tertentu (misalnya, warna merah).
- **Webots API**: Webots digunakan untuk mengontrol robot dan kamera, dan hasil citra diproses dalam simulasi.

7. Implementasi Penggunaan Kamera Bola pada Robot menggunakan Webots

Penggunaan kamera bola di Webots memungkinkan pengamatan 360 derajat untuk robot. Ini sangat berguna untuk navigasi atau pemetaan.

Kode Penggunaan Kamera Bola:

```
from controller import Robot, Camera

# Inisialisasi robot dan kamera bola
robot = Robot()
camera = robot.getCamera("camera_ball")
camera.enable(32)

# Loop utama
while robot.step(32) != -1:
    image = camera.getImage()

    # Analisis citra atau deteksi objek dengan teknik yang sesuai
    pass
```

Penjelasan:

- **Kamera Bola**: Kamera bola memberikan tampilan 360 derajat, sehingga sangat berguna untuk memantau lingkungan sekitar robot.
- **Analisis Citra**: Citra dari kamera bola kemudian bisa dianalisis untuk navigasi atau deteksi objek.

Analisis:

- **Kelebihan**: Menyediakan pandangan penuh di sekitar robot, yang sangat bermanfaat untuk navigasi dan deteksi objek.
- **Keterbatasan**: Memerlukan lebih banyak pemrosesan citra dan algoritma yang efisien untuk mendapatkan hasil yang baik dari gambar 360 derajat.