Analisis Pertama: Webots Example Visual Tracking

Tujuan

Webots untuk mensimulasikan robot yang dapat melacak objek visual menggunakan kamera. Kamera robot menangkap video secara real-time, kemudian OpenCV digunakan untuk memproses frame demi mendeteksi dan melacak objek berdasarkan warna atau pola tertentu.

Struktur

1. Folder utama: Berisi file konfigurasi Webots seperti .wbt (scene robotika), dan kode Python untuk pengolahan citra serta kontrol robot.
2. File Utama Python: Kode implementasi dalam Python, sering disebut controller.py atau nama serupa.

Kode Python Utama

Berikut adalah analisis rinci kode Python yang digunakan

Pengaturan Awal

```
from controller import Robot, Camera, Motor

import cv2

import numpy as np
```

- Robot, Camera, Motor: Modul dari Webots untuk mengendalikan robot.
- cv2 dan np: OpenCV untuk pengolahan citra, NumPy untuk manipulasi array (digunakan untuk data frame kamera).

Inisialisasi Komponen Robot

```
robot = Robot()

time_step = int(robot.getBasicTimeStep())

camera = robot.getDevice('camera')

camera.enable(time_step)

motor_left = robot.getDevice('left_motor')

motor_right = robot.getDevice('right_motor')
```

```
motor_left.setPosition(float('inf'))
motor_right.setPosition(float('inf'))
motor_left.setVelocity(0.0)
motor_right.setVelocity(0.0)
```

- robot.getDevice: Menghubungkan program Python dengan perangkat robot di simulasi.
- Kamera: Diaktifkan untuk menangkap frame secara real-time.
- Motor: Motor kiri dan kanan diatur untuk mode kecepatan tak terbatas dengan kecepatan awal nol.

Loop Utama untuk Pelacakan Visual

```
while robot.step(time_step) != -1:
    frame = np.frombuffer(camera.getImage(), np.uint8).reshape((camera.getHeight(), camera.getWidth(), 4))
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGRA2BGR)

    lower_bound = np.array([30, 150, 50])  # Warna target (HSV)
    upper_bound = np.array([255, 255, 180])

    mask = cv2.inRange(hsv_frame, lower_bound, upper_bound)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        largest_contour = max(contours, key=cv2.contourArea)
        (x, y), radius = cv2.minEnclosingCircle(largest_contour)
        if radius > 10:
```

```
            motor_left.setVelocity(2.0)
            motor_right.setVelocity(2.0)
        else:
            motor_left.setVelocity(0.5)
            motor_right.setVelocity(0.5)
    else:
        motor_left.setVelocity(0.0)
        motor_right.setVelocity(0.0)
```

- Pengolahan Citra:
    - Frame dari kamera diubah dari BGRA ke HSV.
    - Masking warna dilakukan untuk mendeteksi warna target dalam rentang tertentu.
    - Kontur diekstraksi untuk menemukan area objek target.
- Logika Kontrol:
    - Jika ada kontur, robot menggerakkan kedua motor ke arah objek.
    - Radius objek digunakan untuk menentukan apakah objek sudah cukup dekat.

Menghentikan Robot

```
motor_left.setVelocity(0.0)
motor_right.setVelocity(0.0)
```

Ketika loop berhenti, kecepatan motor diatur menjadi nol untuk menghentikan robot.

Simulasi Hasil

- Robot dapat mengikuti objek yang memiliki warna target (dalam rentang HSV).

- Jika objek terlalu kecil (radius < 10), robot melambat atau berhenti.

Analisis Kedua: Fruit Detection Robot with OpenCV and Webots

Tujuan

Membuat robot dalam simulasi Webots yang dapat mendeteksi buah berdasarkan warna atau fitur visual tertentu. Robot dilengkapi kamera yang mengirimkan data ke program Python untuk pengolahan citra menggunakan OpenCV.

Struktur

1. File Konfigurasi Webots: File .wbt berisi konfigurasi lingkungan, termasuk robot dan objek buah.
2. Kode Python: Mengontrol robot untuk mendeteksi buah dan bergerak ke arahnya.

Kode Python Utama

Berikut analisis rinci kode Python yang digunakan:

Inisialisasi Komponen

```
from controller import Robot, Camera, Motor
import cv2
import numpy as np

robot = Robot()
time_step = int(robot.getBasicTimeStep())

camera = robot.getDevice('camera')
camera.enable(time_step)

motor_left = robot.getDevice('left_motor')
motor_right = robot.getDevice('right_motor')
```

```
motor_left.setPosition(float('inf'))
motor_right.setPosition(float('inf'))
motor_left.setVelocity(0.0)
motor_right.setVelocity(0.0)
```

- Webots Integration:
    - Kamera diaktifkan untuk menangkap frame.
    - Motor robot diatur dalam mode kecepatan tak terbatas.

Fungsi Deteksi Buah

```
def detect_fruit(frame):
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 100, 100])
    upper_red = np.array([10, 255, 255])

    mask = cv2.inRange(hsv_frame, lower_red, upper_red)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        largest_contour = max(contours, key=cv2.contourArea)
        return cv2.minEnclosingCircle(largest_contour)
    return None
```

- Menggunakan rentang HSV untuk mendeteksi warna merah (contoh untuk apel).
- Kontur terbesar ditemukan, dan radius lingkaran minimum ditentukan untuk melacak objek.

Loop Utama

```
while robot.step(time_step) != -1:
```

```
    frame = np.frombuffer(camera.getImage(),
np.uint8).reshape((camera.getHeight(), camera.getWidth(), 4))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGRA2BGR)


    result = detect_fruit(frame)
    if result:
        (x, y), radius = result
        if radius > 10:
            motor_left.setVelocity(2.0)
            motor_right.setVelocity(2.0)
        elif x < frame.shape[1] // 2:
            motor_left.setVelocity(0.5)
            motor_right.setVelocity(1.0)
        else:
            motor_left.setVelocity(1.0)
            motor_right.setVelocity(0.5)
    else:
        motor_left.setVelocity(0.0)
        motor_right.setVelocity(0.0)
```

- Kontrol Robot:
    - Jika buah terdeteksi, robot bergerak maju.
    - Jika objek tidak di tengah, motor diatur untuk menyesuaikan arah.

Analisis Ketiga: Document Scanner Simulation

Tujuan

Mensimulasikan robot yang memindai dokumen menggunakan kamera. OpenCV digunakan untuk mendeteksi tepi dokumen, memperbaiki perspektif, dan menyimpan hasil sebagai gambar.

Struktur

1. File Simulasi Webots: Berisi konfigurasi lingkungan untuk robot.
2. Kode Python: Implementasi deteksi tepi dan transformasi perspektif.

Kode Python Utama

Berikut penjelasan rinci kode Python yang digunakan:

Fungsi Utama untuk Deteksi Tepi

```
def preprocess_image(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edged = cv2.Canny(blurred, 75, 200)
    return edged
```

- Pengolahan Awal:

  Gambar diubah menjadi grayscale, dirapikan dengan Gaussian Blur, lalu deteksi tepi menggunakan Canny.

Transformasi Perspektif

```
def four_point_transform(image, points):
    rect = order_points(points)
    (tl, tr, br, bl) = rect
    width = max(int(np.linalg.norm(br - bl)), int(np.linalg.norm(tr - tl)))
    height = max(int(np.linalg.norm(tr - br)), int(np.linalg.norm(tl - bl)))

    dst = np.array([[0, 0], [width - 1, 0], [width - 1, height - 1], [0, height - 1]], dtype="float32")
    matrix = cv2.getPerspectiveTransform(rect, dst)
```

```
    return cv2.warpPerspective(image, matrix, (width, height))
```

- Transformasi:

  Koordinat tepi dokumen digunakan untuk memperbaiki perspektif agar dokumen sejajar.

Loop Simulasi

```
while robot.step(time_step) != -1:
    frame = np.frombuffer(camera.getImage(), np.uint8).reshape((camera.getHeight(), camera.getWidth(), 4))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGRA2BGR)
    edged = preprocess_image(frame)
    contours, _ = cv2.findContours(edged, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)


    if contours:
        largest_contour = max(contours, key=cv2.contourArea)
        approx = cv2.approxPolyDP(largest_contour, 0.02 * cv2.arcLength(largest_contour, True), True)
        if len(approx) == 4:
            scanned = four_point_transform(frame, approx.reshape(4, 2))
            cv2.imshow("Scanned", scanned)
            cv2.waitKey(1)
```

- Hasil:
  - Deteksi dokumen berhasil jika kontur memiliki 4 sisi.
  - Perspektif dokumen diperbaiki dan ditampilkan.