Analisa Program Webots Week 10

Rafli Limandijaya/1103210243

Visual tracking

Sebuah robot memiliki camera yang terintegrasi dengan OpenCV yang terus mengikuti sebuah bola yang berputar mengelilinginya.

Komponen utama pada program :

Kode ini bertujuan untuk mengontrol robot agar mengikuti bola merah menggunakan algoritma proportional control (P controller). Berikut adalah penjelasan lengkap mengenai cara kerja kode:

1. Inisialisasi Komponen

Robot:

robot = Robot()

timestep = int(robot.getBasicTimeStep())

Robot diinisialisasi dengan mengambil timestep dasar dari simulator Webots untuk menentukan frekuensi langkah.

Kamera:

camera = robot.getDevice('camera')

camera.enable(timestep)

Kamera diaktifkan untuk menangkap gambar yang akan digunakan dalam deteksi objek (bola merah).

Motor:

motor_left = robot.getDevice('left wheel motor')

motor_right = robot.getDevice('right wheel motor')

motor_left.setPosition(float('inf'))

motor_right.setPosition(float('inf'))

motor_left.setVelocity(0)

motor_right.setVelocity(0)

Motor roda kiri dan kanan diatur pada posisi tak terbatas (infinite), memungkinkan robot untuk bergerak secara kontinu. Kecepatan awal diatur menjadi 0.

2. Logika Utama (Loop)

Pada loop utama, gambar dari kamera diproses untuk mendeteksi posisi bola merah, dan motor dikontrol berdasarkan posisi tersebut.

a. Mengambil dan Mengonversi Gambar

img = np.frombuffer(camera.getImage(), dtype=np.uint8).reshape((camera.getHeight(), camera.getWidth(), 4))

img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

camera.getImage(): Mengambil gambar dari kamera dalam format RGBA.

np.frombuffer: Mengonversi gambar menjadi array NumPy.

cv2.cvtColor: Mengonversi gambar ke ruang warna HSV (Hue, Saturation, Value), yang lebih mudah untuk segmentasi berdasarkan warna.

b. Segmentasi Warna

mask = cv2.inRange(img, np.array([50, 150, 0]), np.array([200, 230, 255]))

cv2.inRange: Membuat mask berdasarkan rentang warna bola merah di ruang warna HSV.

Rentang warna ini perlu disesuaikan dengan warna bola di simulasi.

Nilai HSV [50, 150, 0] hingga [200, 230, 255] menentukan batas atas dan bawah warna.

c. Deteksi Kontur

contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

largest_contour = max(contours, key=cv2.contourArea)

largest_contour_center = cv2.moments(largest_contour)

center_x = int(largest_contour_center['m10'] / largest_contour_center['m00'])

cv2.findContours: Mendeteksi kontur pada gambar hasil segmentasi.

max(contours, key=cv2.contourArea): Memilih kontur terbesar, diasumsikan sebagai bola merah.

cv2.moments: Menghitung momen kontur untuk menentukan pusat (center) dari bola.

d. Hitung Kesalahan Posisi

error = camera.getWidth() / 2 - center_x

Posisi tengah gambar dihitung dengan camera.getWidth() / 2.

Posisi bola di gambar (center_x) dibandingkan dengan tengah gambar untuk menentukan error (jarak horizontal bola dari tengah gambar).

e. P Controller

motor_left.setVelocity(- error * P_COEFFICIENT)

motor_right.setVelocity(error * P_COEFFICIENT)

Proportional Control digunakan untuk menentukan kecepatan motor berdasarkan nilai kesalahan (error).

Jika bola berada di tengah gambar, kesalahan = 0, sehingga motor berhenti berbelok.

Jika bola bergerak ke kiri/kanan, kecepatan motor kiri dan kanan disesuaikan untuk mengarahkan robot menuju bola.

2. Fruit Detection

Mulai dari sini, saya akan menjelaskan saja cara kerja dari beberapa bagian kode yang penting.

Cara kerja kode :

1. Mengaktifkan Kamera

Kamera diaktifkan untuk menangkap gambar dan mengenali objek:

```
camera = supervisor.getDevice("camera")
camera.enable(2 * TIME_STEP)
camera.recognitionEnable(2 * TIME_STEP)
```

2. Mengambil Gambar Kamera

Data gambar diambil dalam format BGRA dan akan diproses:

```
image_data = camera.getImage()
width = camera.getWidth()
height = camera.getHeight()
```

3. Pemrosesan Gambar Menggunakan OpenCV

Gambar dikonversi ke format BGR, lalu ke grayscale, dan mendeteksi tepi:

```
edges = process_image_with_opencv(image_data, width, height)
```

4. Mendeteksi Objek

Kamera mendeteksi jumlah dan informasi objek dalam pandangannya:

```
number_of_objects = camera.getRecognitionNumberOfObjects()
objects = camera.getRecognitionObjects()
```

5. Mengekstrak Model Objek

Model objek yang terdeteksi diperiksa untuk menentukan jenis buah:

```
if number_of_objects > 0:
    fruit = objects[0].getModel()
    if fruit[0] == 'a':  # Karakter pertama 'a' untuk apel
        model = 1
    else:
        model = 0
```

6. Logika Robot Berdasarkan Model

Jenis buah memengaruhi tindakan robot, seperti membuka gripper atau memindahkan lengan:

```
if state == WAITING:
   if distance_sensor.getValue() < 500:
      state = PICKING
      if model == 1:
         apple += 1
      else:
         orange += 1
```

7. Menampilkan Jumlah Buah

Jumlah apel dan jeruk yang dikenali diperbarui dan ditampilkan:

```
strP = f"Oranges: {orange}"
supervisor.setLabel(0, strP, 0.45, 0.96, 0.06, 0x5555ff, 0, "Lucida Console")

strP = f"Apples: {apple}"
supervisor.setLabel(1, strP, 0.3, 0.96, 0.06, 0x5555ff, 0, "Lucida Console")
```

3. Document Scanner

Pada simulasi ini, nampaknya kotak-kotak kehilangan teksturnya sehingga simulasi tidak berjalan dengan baik. Dibawah ini adalah penjelasan untuk kodenya.

Cara kerja kode

- Import Library dan Konstanta:
    - Library Standar dan Eksternal:
        - cv2 dan numpy digunakan untuk manipulasi gambar.
        - itertools.count digunakan untuk membuat counter otomatis.
        - controller dari Webots mengontrol robot dan perangkat terkait.
    - Konstanta:
        - TIME_STEP: Interval langkah simulasi.
        - HSV_LOW_RANGE dan HSV_UP_RANGE: Rentang warna untuk segmentasi dalam ruang warna HSV.

    - SAVE_TO_DISK: Flag untuk menyimpan gambar ke disk (default: False).

- Fungsi Pendukung:
    - counter(): Menghasilkan angka bertambah secara otomatis (untuk penamaan file gambar).
    - save_image(image): Menyimpan gambar ke file dengan nama berurutan.
    - initialize(): Menginisialisasi robot, kamera, dan layar tampilan (display) Webots.
    - webots_image_to_numpy(im, h, w): Mengonversi gambar Webots menjadi array Numpy.
    - display_numpy_image(im, display, display_width, display_height):
        - Menampilkan gambar Numpy di layar tampilan Webots.
- Fungsi Utama:
    - Segmentasi Warna:
        - Fungsi segment_by_color digunakan untuk mendeteksi warna dokumen berdasarkan rentang HSV.
    - Pemrosesan Dokumen:
        - Fungsi get_warped_document digunakan untuk memperbaiki perspektif dokumen sehingga menjadi lurus.
        - Fungsi resize_and_letter_box mengubah ukuran gambar dokumen agar sesuai dengan layar tampilan.
    - Error Handling:
        - Jika dokumen tidak terdeteksi, layar tampilan diisi dengan gambar hitam.
- Loop Utama:
    - Pada setiap langkah simulasi (robot.step(TIME_STEP)):
        - Gambar diambil dari kamera dan dikonversi ke format Numpy.
        - Gambar diproses untuk mendeteksi dokumen:
            - Segmentasi warna dilakukan untuk membuat mask dari dokumen.

        - Perspektif dokumen diperbaiki menggunakan get_warped_document.
        - Gambar dokumen ditampilkan di layar tampilan Webots.
    - Jika SAVE_TO_DISK = True, gambar asli disimpan ke disk.

- Akhir Simulasi:
    - robot.cleanup() digunakan untuk membersihkan sumber daya saat simulasi selesai.