Nama : Muhammad Husain Kaasyiful Ghitha

NIM : 1103220047

Kelas TK-45-G09

Robotika dan Sistem Cerdas

Analisis Tugas Robotika Week 12

Pada tugas ini, dilaksanakan beberapa penerapan ekstraksi informasi (Information Extraction),, menggunakan bahasa Python. Simulasi-simulasi yang diprogram adalah sebagai berikut:

- Ekstraksi Garis dengan Hough Transform
- Template Matching untuk Deteksi Objek.
- Pembuatan Pyramid Gambar.
- Deteksi Lingkaran Menggunakan Hough Transform.
- Ekstraksi Warna Dominan pada Gambar.
- Deteksi Kontur pada Gambar.

Selain memrogram kode Python untuk simulasi tersebut, terdapat juga simulasi yang berfokus pada implementasinya pada robot menggunakan simulator Webots. Simulasi yang dilakukan pada Webots tersebut adalah Lidar Data Extraction and Obstacle Detection.

Berikut adalah pembahasan mengenai kode Python dan simulasi-simulasi yang dilakukan.

1. Membuat kode Python menggunakan OpenCV untuk simulasi mengenai Information Extraction.

   Pada kode simulasi ini, penjelasan fungsi-fungsi yang dimuat di program adalalah sebagai berikut:

   a. Ekstraksi Garis dengan Hough Transform.

      Ekstraksi Garis menggunakan Hough Transform adalah metode untuk mendeteksi garis dalam gambar dengan mengubah titik-titik di ruang kartesius menjadi kurva di ruang parameter polar (ρ, θ), di mana ρ adalah jarak garis dari asal, dan θ\thetaθ

adalah sudut garis terhadap sumbu x. Titik-titik yang sejajar pada sebuah garis di ruang gambar akan bertemu di satu titik dalam ruang Hough. Prosesnya melibatkan edge detection (seperti Canny), transformasi ke ruang Hough, dan penggunaan matriks accumulator untuk mengidentifikasi kombinasi (ρ,θ) dengan akumulasi tertinggi sebagai garis yang terdeteksi. Metode ini efektif untuk mendeteksi garis meskipun ada noise, tetapi membutuhkan komputasi besar untuk gambar beresolusi tinggi.

Kode pada program menggunakan Hough Transform untuk mendeteksi garis pada gambar. Gambar dimuat dalam mode grayscale, lalu tepi gambar dideteksi menggunakan algoritma Canny dengan threshold 50 dan 150. Hough Transform diterapkan pada gambar tepi untuk mendeteksi garis, dengan parameter resolusi ruang Hough (ρ sebesar 1 dan θ sebesar 1°) serta threshold akumulator sebesar 100. Jika garis terdeteksi, setiap garis direpresentasikan dalam parameter polar (ρ,θ), kemudian dihitung koordinat dua titik ekstremnya untuk digambar sebagai garis hijau pada gambar asli yang diubah ke mode BGR. Hasilnya ditampilkan dalam dua subplot: gambar asli dan gambar dengan garis-garis yang terdeteksi.

Hasil output simulasi dengan gambar berupa garis pada plot grafik menunjukkan gambar asli di sebelah kiri dengan pola garis-garis diagonal hitam dalam sebuah plot grafik. Di sebelah kanan, hasil deteksi garis menggunakan transformasi Hough ditampilkan, di mana sebagian besar garis pada gambar berhasil diidentifikasi dan digambar ulang dengan warna hijau. Ini membuktikan bahwa algoritma deteksi tepi dan transformasi Hough bekerja dengan baik dalam mengenali garis lurus pada pola yang teratur seperti pada gambar ini.

b. Template Matching untuk Deteksi Objek.

Template Matching adalah metode sederhana untuk mendeteksi objek dalam gambar dengan mencocokkan template (sub-gambar) ke dalam gambar target. Prosesnya melibatkan pergeseran template ke setiap posisi dalam gambar dan menghitung tingkat kecocokan pada setiap lokasi menggunakan metode seperti korelasi atau perhitungan perbedaan intensitas. Hasilnya adalah peta nilai kecocokan, di mana nilai maksimum menunjukkan lokasi terbaik untuk pencocokan. Metode ini cocok untuk mendeteksi objek dengan bentuk dan ukuran tetap tetapi kurang efektif jika objek memiliki rotasi, skala berbeda, atau ada noise signifikan.

Kode di atas menggunakan Template Matching untuk mendeteksi objek dalam gambar dengan langkah-langkah sebagai berikut: Gambar utama dan template dimuat, lalu template divalidasi untuk memastikan ukurannya lebih kecil dari gambar utama. Gambar utama dikonversi ke grayscale, dan fungsi cv2.matchTemplate digunakan untuk mencocokkan template ke gambar menggunakan metode korelasi ternormalisasi (cv2.TM_CCOEFF_NORMED). Lokasi kecocokan terbaik diperoleh melalui cv2.minMaxLoc, dan area deteksi ditandai dengan kotak hijau serta label "Sesuai Template". Hasil akhirnya ditampilkan dalam tiga subplot: gambar utama, template, dan gambar utama dengan area deteksi. Kode juga menangani error menggunakan try-except, misalnya jika template lebih besar dari gambar utama.

Output program pada percobaan menunjukkan bahwa algoritma Template Matching berhasil mendeteksi objek pada gambar utama. Pada gambar hasil, area di mana template cocok dengan gambar utama ditandai dengan kotak hijau, dan label ditambahkan di atasnya. Ini membuktikan bahwa metode ini dapat secara akurat menemukan posisi objek dalam gambar dengan mencocokkan pola template dengan area pada gambar utama.

c. Pembuatan Gaussian Pyramid Gambar.

Gaussian Pyramid adalah teknik untuk merepresentasikan gambar dalam berbagai skala dengan mengurangi resolusi secara bertahap menggunakan filter Gaussian. Prosesnya dimulai dengan gambar asli sebagai level dasar (level 0), kemudian setiap level berikutnya dibuat dengan mengaburkan gambar menggunakan filter Gaussian dan mengurangi ukuran (downsampling) dengan menghilangkan setiap piksel kedua secara horizontal dan vertikal. Hasilnya adalah serangkaian gambar dengan resolusi yang lebih kecil tetapi mempertahankan struktur inti dari gambar asli. Gaussian Pyramid digunakan dalam berbagai aplikasi seperti blending gambar, deteksi objek multi-skala, dan analisis citra pada skala yang berbeda.

Kode di atas membuat Gaussian Pyramid dengan menghasilkan tiga level dari gambar input. Gambar dimuat terlebih dahulu menggunakan OpenCV, lalu setiap level pyramid dibuat dengan fungsi cv2.pyrDown, yang mengaburkan gambar menggunakan filter Gaussian dan melakukan downsampling (mengurangi resolusi). Hasil dari setiap level ditambahkan ke dalam list pyramid gambar. Terkhir, setiap level dari Gaussian Pyramid ditampilkan dalam subplot menggunakan Matplotlib,

dengan Level 0 sebagai gambar asli, dan level-level berikutnya menunjukkan gambar dengan resolusi yang lebih rendah secara bertahap.

Hasil simulasi menunjukkan empat level Gaussian Pyramid dari gambar yang diunggah. Gabar yang dilabel Level 0 adalah gambar asli dengan resolusi penuh, sementara Level 1 hingga Level 3 menunjukkan gambar dengan resolusi yang semakin rendah. Setiap level gambar lebih kecil dan detailnya berkurang, menghasilkan representasi sederhana dari gambar asli.

d. Deteksi Lingkaran Menggunakan Hough Transform.

Deteksi lingkaran menggunakan Hough Transform adalah metode untuk menemukan lingkaran dalam gambar dengan merepresentasikan lingkaran dalam bentuk parameter ruang Hough. Sebuah lingkaran dapat didefinisikan oleh tiga parameter: pusat koordinatnya (x, y) dan jari-jari r. Metode ini memindai gambar untuk mendeteksi piksel-piksel tepi (misalnya, dengan algoritma Canny), kemudian memetakan kemungkinan pusat dan jari-jari ke ruang parameter.

Pada program simulasi, Fungsi cv2.HoughCircles digunakan untuk mengidentifikasi lingkaran pada gambar dengan parameter seperti resolusi akumulator, jarak minimum antar lingkaran, dan threshold untuk deteksi. Gambar dimuat dalam mode berwarna, lalu dikonversi menjadi grayscale untuk mempermudah proses deteksi. Fungsi cv2.HoughCircles digunakan dengan metode cv2.HOUGH_GRADIENT, di mana parameter seperti resolusi akumulator (dp), jarak minimum antar lingkaran (minDist), dan sensitivitas deteksi (param1 untuk threshold Canny dan param2 untuk sensitivitas deteksi lingkaran) disesuaikan. Selain itu, batas ukuran radius lingkaran yang dapat dideteksi ditentukan melalui minRadius dan maxRadius. Jika lingkaran terdeteksi, setiap lingkaran digambar pada gambar asli dengan lingkaran hijau untuk tepinya dan lingkaran merah untuk pusatnya. Hasil akhirnya ditampilkan menggunakan matplotlib dalam dua subplot: gambar asli dan gambar dengan lingkaran yang terdeteksi.

Hasil output menunjukkan bahwa metode Hough Transform berhasil mendeteksi lingkaran pada plot grafik. Pada gambar hasil, lingkaran hijau digambar mengelilingi semua lingkaran yang ada, menandakan batas lingkaran yang terdeteksi, sementara titik merah menunjukkan pusat lingkaran. Ini membuktikan algoritma bekerja dengan baik dalam mengenali bentuk lingkaran, bahkan pada gambar dengan lingkaran yang saling bertindihan.

e. Ekstraksi Warna Dominan pada Gambar.

Ekstraksi warna dominan pada gambar adalah proses untuk menentukan warna-warna yang paling sering muncul dalam sebuah gambar, biasanya menggunakan teknik clustering seperti k-Means Clustering. Proses dimulai dengan mengonversi gambar ke dalam ruang warna (seperti RGB atau HSV) dan meratakan piksel menjadi array 2D. Algoritma k-Means Clustering kemudian diterapkan untuk mengelompokkan piksel berdasarkan nilai warna, dengan setiap kluster merepresentasikan satu warna dominan. Setelah kluster ditemukan, rata-rata dari setiap kluster dihitung sebagai representasi warna dominan, dan frekuensi relatifnya dapat digunakan untuk memprioritaskan warna. Teknik ini digunakan dalam berbagai aplikasi seperti analisis gambar, rekomendasi warna, atau filter estetika pada foto.

Kode pada program simulasi melakukan ekstraksi warna dominan pada gambar menggunakan algoritma K-Means Clustering. Gambar dimuat menggunakan OpenCV, dikonversi dari format BGR (input OpenCV) ke RGB, lalu diratakan menjadi array 2D di mana setiap baris merepresentasikan satu piksel dalam format RGB. Algoritma K-Means diterapkan dengan jumlah kluster warna k = 5, sehingga menghasilkan lima warna dominan yang disimpan dalam kmeans.cluster_centers_. Warna dominan ini kemudian dikonversi ke tipe integer 8-bit untuk ditampilkan. Gambar asli ditampilkan di subplot pertama, diikuti oleh masing-masing warna dominan dalam subplot lainnya, di mana setiap warna divisualisasikan sebagai kotak berwarna dengan nilai normalisasi.

K-Means Clustering berhasil mengekstraksi 5 warna dominan dari gambar asli berupa pemandangan alam pada percobaan simulasi yang dilakukan. Warna dominan yang dihasilkan, seperti hijau kekuningan, hijau gelap, dan biru terang, merepresentasikan elemen utama dalam gambar seperti rerumputan, pohon, dan langit. Warna-warna ini ditampilkan dalam bentuk kotak dengan label "Color 1" hingga "Color 5", memberikan gambaran palet warna utama pada gambar. Analisis ini berguna untuk memahami komposisi warna dalam desain atau citra visual.

f. Deteksi Kontur pada Gambar.

Deteksi kontur pada gambar menggunakan Python dan OpenCV adalah proses untuk menemukan dan mewakili batas-batas objek dalam gambar. Proses ini

dimulai dengan mengubah gambar menjadi grayscale dan kemudian melakukan thresholding atau edge detection (seperti algoritma Canny) untuk mendapatkan tepi objek. Fungsi cv2.findContours digunakan untuk mendeteksi kontur, yang mengembalikan daftar koordinat titik-titik tepi untuk setiap kontur yang ditemukan. Parameter dalam fungsi ini memungkinkan pengguna memilih metode aproksimasi kontur, seperti menyimpan semua titik (CHAIN_APPROX_NONE) atau hanya titik-titik penting (CHAIN_APPROX_SIMPLE). Kontur yang ditemukan dapat digambar di atas gambar asli menggunakan cv2.drawContours untuk divisualisasikan..

Hasil output menunjukkan bahwa algoritma telah berhasil mendeteksi kontur pada gambar contoh berupa gambar shuttlecock (kok). Kontur luar kok ditandai dengan jelas menggunakan garis hijau, sementara beberapa garis dalam, seperti detail jaring, juga terdeteksi. Deteksi ini menunjukkan kemampuan algoritma untuk menangkap elemen-elemen penting dalam gambar. Jika hanya ingin fokus pada kontur luar kok, filter tambahan untuk mengabaikan kontur kecil dapat diterapkan. Secara keseluruhan, hasil ini cukup baik untuk analisis objek utama.

2. Simulasi Webots yang ada Lidar Data Extraction and Obstacle Detection.

   Ekstraksi Data dan Rintangan menggunakan Lidar (Light Detection and Ranging) adalah proses penting dalam aplikasi robotika untuk memahami lingkungan sekitar dan menghindari rintangan. Lidar bekerja dengan memancarkan laser dan mengukur waktu pantulan kembali untuk menghitung jarak dari objek. Data Lidar berupa kumpulan titik (point cloud) yang merepresentasikan jarak dan sudut objek di sekitar robot. Dalam proses ekstraksi, data ini diolah untuk mengidentifikasi fitur-fitur penting, seperti posisi rintangan atau peta lingkungan. Algoritma seperti clustering (misalnya DBSCAN) dapat digunakan untuk mengelompokkan data titik menjadi objek yang terpisah, sementara teknik filtering digunakan untuk menghapus noise. Untuk deteksi rintangan, jarak titik Lidar dianalisis untuk menentukan apakah ada objek dalam zona aman robot. Informasi ini sering digunakan dalam navigasi otonom, perencanaan jalur, dan sistem penghindaran tabrakan. Kombinasi Lidar dengan sensor lain, seperti kamera atau GPS, dapat meningkatkan akurasi dalam pengambilan keputusan robot.

   Kode Controller yang diberikan mengimplementasikan kontrol robot di Webots untuk mengekstraksi data Lidar dan mendeteksi rintangan. Robot diinisialisasi bersama dengan perangkat-perangkatnya dengan fungsi getDevice(), termasuk perangkat Lidar

untuk pemetaan lingkungan, sensor jarak ultrasonik untuk deteksi jarak dekat, dan motor untuk menggerakkan roda kiri dan kanan. Sensor-sensor diaktifkan dengan waktu langkah simulasi yang ditentukan oleh konstanta TIME_STEP. Motor diatur dalam mode kecepatan tanpa batas agar robot dapat bergerak secara kontinu dengan kecepatan yang dapat disesuaikan.

Pada fungsi extract_lidar_data(), objek Lidar digunakan untuk membaca data jarak dari objek di sekitarnya menggunakan metode getRangeImage(), yang menghasilkan array jarak dari berbagai sudut pandang robot. Sebagai contoh, data pertama yang diambil ditampilkan pada konsol untuk debugging. Selain itu, data sensor jarak ultrasonik dibaca menggunakan fungsi read_distance_sensors(), yang memberikan nilai jarak pada sisi kiri dan kanan robot untuk mendeteksi rintangan secara lokal.

Fungsi utama dalam logika kontrol adalah compute_speeds(), yang menghitung kecepatan roda berdasarkan nilai sensor jarak dengan menggunakan matriks koefisien empiris. Koefisien ini menentukan respons motor terhadap nilai jarak sensor, seperti memperlambat roda kanan jika ada rintangan di sisi kiri untuk menghindari tabrakan. Kecepatan dasar ditambahkan ke hasil perhitungan untuk memastikan robot terus bergerak maju sambil melakukan koreksi arah sesuai dengan data sensor.

Dalam loop utama, data Lidar dan sensor jarak dibaca secara berkala dengan memanggil extract_lidar_data(). Kecepatan roda dihitung berdasarkan data sensor yang didapat dari read_distance_sensors(), lalu diterapkan ke motor kiri dan kanan menggunakan fungsi setVelocity(). Loop berjalan terus hingga simulasi dihentikan. Implementasi ini memungkinkan robot mendeteksi rintangan dan menyesuaikan arah gerakannya secara dinamis, membuatnya cocok untuk navigasi otonom sederhana dalam lingkungan dengan rintangan.

Hasil dari simulasi yang dilakukan adalah robot mampu mengenali objek-objek yang terletak di dalam lingkungan, juga mengenali batas-batas lingkungan yang diberikan. Robot mengumpulkan jarak-jarak dari berbagai sudut robot menggunakan lidar, yang ditampilkan pada simulasi sebagai garis sinar yang menyebar dari robot ke lingkungan, terhenti oleh jarak atau rintangan. Jarak yang dikeluarkan oleh lidar tersebut dapat digunakan untuk mendeteksi rintangan dan menghindari tabrakan.