Ade Tirta Rahmat Hidayat – 1103203212 – TK45G06

Hasil Analisis Percobaan Information Extraction dan LIDAR

Link Google Colab:
https://colab.research.google.com/drive/10cPyQ0Jcm6vnTAPjYZXQvVqJfl3Y78Ti?usp=sharing

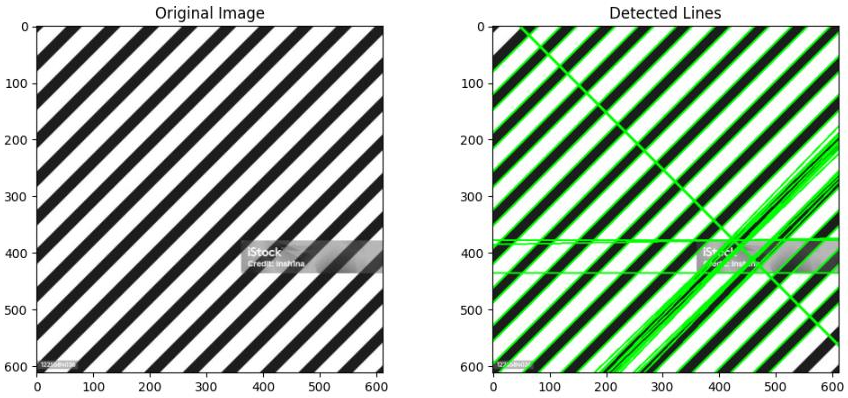
A. Ekstraksi Garis dengan Hough Transform

Hough Transform adalah teknik dalam pemrosesan gambar yang digunakan untuk mendeteksi garis atau bentuk geometris lainnya dalam sebuah gambar. Adapun pada percobaan yang dilakukan, langkah-langkah dalam implementasi ekstrasi garis ini meliputi:

1. Unggah dan Baca Gambar
   Gambar dibaca dalam skala abu-abu menggunakan cv2.IMREAD_GRAYSCALE. Skala abu-abu mempermudah deteksi tepi karena mengurangi dimensi data (hanya satu channel).
2. Deteksi Tepi dengan Canny
   Metode *Canny Edge Detection* digunakan untuk mendeteksi tepi dalam gambar. Parameter pertama dan kedua (50, 150) adalah nilai ambang bawah dan atas untuk mendeteksi tepi. Hasilnya adalah sebuah gambar biner yang hanya menyimpan informasi tepi (0 untuk latar belakang, 255 untuk tepi).
3. Deteksi Garis Menggunakan Hough Transform
   Fungsi cv2.HoughLines digunakan untuk mendeteksi garis. Adapun parameternya meliputi:
   - edges: Gambar tepi hasil dari Canny.
   - 1: Resolusi jarak dalam piksel.
   - np.pi / 180: Resolusi sudut dalam radian (1 derajat = $\pi/180$ radian).
   - 100: Ambang batas jumlah akumulator yang harus dilampaui agar garis dianggap terdeteksi.

   Output dari cv2.HoughLines adalah sekumpulan parameter (rho, theta) yang mendefinisikan garis dalam koordinat polar:
   - rho: Jarak dari asal ke garis.
   - theta: Sudut antara sumbu x dan garis.
4. Menggambar Garis pada Gambar Asli
   Untuk setiap (rho, theta) dari hasil Hough Transform, rho dan theta dikonversi menjadi koordinat kartesian (x1, y1) dan (x2, y2) untuk menggambar garis menggunakan fungsi cv2.line. Garis-garis ini digambar pada gambar asli dengan warna hijau (0, 255, 0).

Hasil dari percobaan tersebut yaitu:



Hasil Analisis:

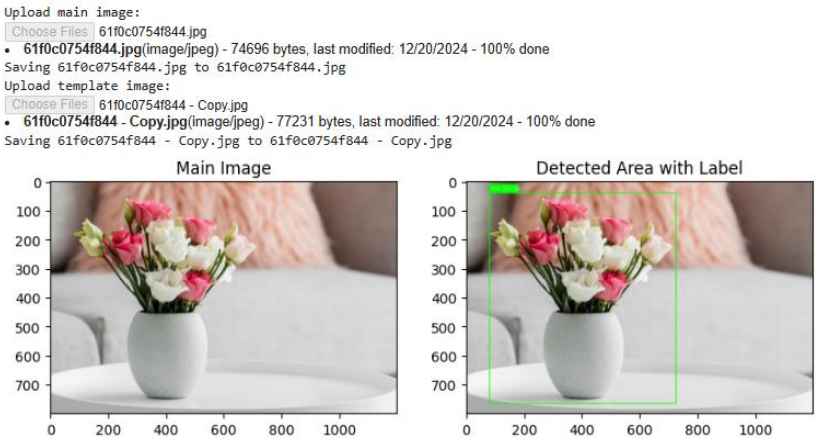
- Gambar Kiri (Original Image): Merupakan gambar awal yang hanya menampilkan pola garis diagonal hitam dan putih.
- Gambar Kanan (Detected Lines): Semua garis diagonal pada gambar berhasil dideteksi dan digambar dengan warna hijau. Terlihat bahwa algoritma juga mendeteksi beberapa garis horizontal dan vertikal tambahan. Hal ini bisa terjadi karena pola gambar memiliki elemen noise (seperti watermark) dan ambang batas (*threshold*) dalam deteksi Hough Transform cukup rendah sehingga lebih banyak garis terdeteksi.

B. Template Matching untuk Deteksi Objek

Template Matching adalah metode dalam pengolahan gambar digital yang digunakan untuk menemukan lokasi tertentu dari objek (template) dalam gambar utama. Adapun langkah-langkah yang dilakukan yaitu:

1. Validasi Ukuran Template:
   Fungsi validate_template_size memastikan bahwa ukuran template lebih kecil dari gambar utama. Hal ini penting karena Template Matching tidak dapat dilakukan jika template lebih besar.
2. Unggah dan Baca Gambar:
   Gambar utama diunggah dan dibaca dalam format berwarna (RGB) sedangkan template diunggah dan dibaca dalam skala abu-abu (grayscale). Hal ini dilakukan untuk mengurangi dimensi data yang diproses dan menyederhanakan perhitungan.
3. Konversi Gambar Utama ke Grayscale:
   Gambar utama juga dikonversi menjadi grayscale menggunakan cv2.cvtColor agar sesuai dengan format template.
4. Template Matching:
   Fungsi cv2.matchTemplate digunakan untuk membandingkan template dengan gambar utama menggunakan metode cv2.TM_CCOEFF_NORMED. Metode ini mengembalikan peta hasil (result matrix) yang menunjukkan tingkat kecocokan antara template dan setiap bagian gambar utama. Koordinat titik dengan nilai kecocokan maksimum (max_val) diperoleh menggunakan cv2.minMaxLoc.
5. Menggambar Kotak Deteksi:
   Berdasarkan lokasi maksimum (max_loc), kode menggambar kotak hijau pada gambar utama untuk menandai area yang cocok dengan template.

Berdasarkan langkah-langkah tersebut, diperoleh hasil dari percobaan ini yaitu:



Hasil Analisis:

- Gambar Kiri (Main Image): Gambar utama yang berisi "vas bunga" berada di tengah latar belakang ruang tamu.
- Gambar Kanan (Detected Area with Label): Metode Template Matching berhasil mendeteksi lokasi vas bunga dengan akurasi tinggi. Kotak hijau terlihat mengelilingi vas bunga dengan posisi yang tepat, menunjukkan kecocokan yang tinggi antara template dan gambar utama.

C. Pembuatan Pyramid Gambar

Gaussian Pyramid adalah representasi bertingkat dari gambar yang diperoleh dengan mengurangi resolusi gambar secara iteratif. Adapun langkah-langkahnya meliputi:

1. Mengunggah dan Membaca Gambar
   Gambar diunggah menggunakan fungsi files.upload() dan dibaca dengan cv2.imread() dalam format BGR.
2. Pembuatan Gaussian Pyramid
   Level 0 adalah gambar asli. Gambar diproses iteratif menggunakan cv2.pyrDown() untuk menghasilkan versi gambar dengan resolusi yang lebih rendah. Setiap iterasi mengurangi ukuran gambar menjadi setengah dari dimensi sebelumnya (baik lebar maupun tinggi), sambil mempertahankan struktur gambar.
3. Visualisasi Pyramid

Adapun hasil dari percobaan tersebut yaitu:

Hasil Analisis:

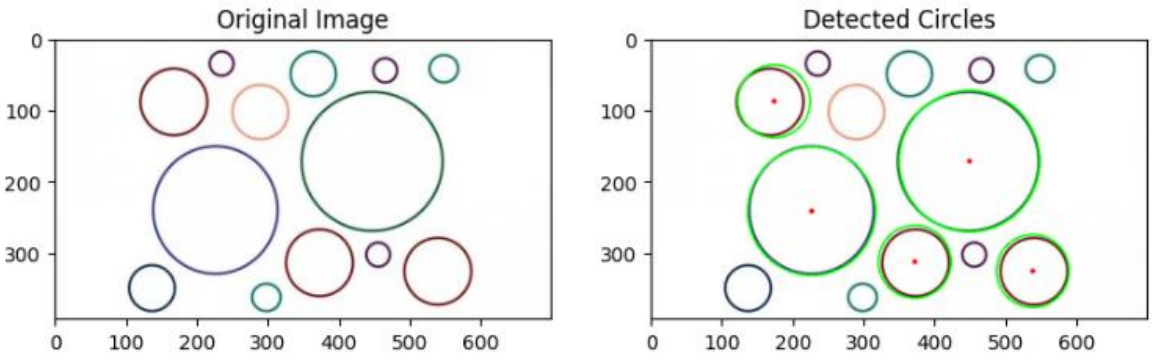
- Level 0: Ini adalah gambar asli dengan resolusi penuh. Semua detail gambar, termasuk tekstur dan elemen kecil, terlihat dengan jelas.
- Level 1: Gambar mengalami pengurangan resolusi pertama. Beberapa detail halus mulai menghilang, namun struktur umum masih terlihat jelas. Selain itu, ukuran gambar menjadi setengah dari ukuran gambar asli.
- Level 2: Resolusi berkurang lebih lanjut, sehingga detail kecil seperti tekstur rumput dan air menjadi kurang terlihat. Elemen besar seperti garis sungai dan gunung tetap terlihat.
- Level 3: Ini adalah level dengan resolusi terendah. Gambar mulai terlihat buram, dan hanya struktur global seperti bentuk sungai dan warna dominan yang terlihat.

D. Deteksi Lingkaran menggunakan Hough Transform

Hough Circle Transform adalah algoritma yang digunakan untuk mendeteksi lingkaran pada gambar berdasarkan parameter seperti pusat dan radius. Adapun langkah-langkahnya meliputi:

1. Unggah dan Baca Gambar
   Gambar diunggah menggunakan files.upload() dan dibaca menggunakan cv2.imread() dalam format BGR (berwarna).
2. Konversi ke Grayscale
   Gambar diubah menjadi grayscale menggunakan cv2.cvtColor(). Hal ini penting karena deteksi lingkaran lebih efektif pada gambar grayscale.
3. Hough Circle Transform
   Fungsi cv2.HoughCircles() digunakan untuk mendeteksi lingkaran. Parameter utama yang memengaruhi deteksi adalah:
   - dp: Resolusi akumulator Hough relatif terhadap gambar asli (1.2 dalam kode ini).
   - minDist: Jarak minimum antar pusat lingkaran (50 piksel).
   - param1: Ambang batas untuk deteksi tepi (Canny edge detection).
   - param2: Threshold akumulator Hough untuk validasi lingkaran.
   - minRadius dan maxRadius: Batasan radius lingkaran untuk menghindari deteksi yang salah.
4. Menggambar Lingkaran yang Terdeteksi
   Lingkaran yang terdeteksi digambar pada gambar asli menggunakan cv2.circle():
   - Lingkaran luar digambar dengan warna hijau.
   - Titik pusat lingkaran digambar dengan warna merah.
5. Visualisasi Hasil

Berdasarkan langkah-langkah tersebut, didapatkan hasil dari percobaan ini yaitu:



Hasil Analisis:

- Original Image: Gambar asli menunjukkan kumpulan lingkaran dengan berbagai ukuran dan lokasi.
- Detected Circles: Gambar hasil menunjukkan lingkaran-lingkaran yang telah terdeteksi, dengan lingkaran luar (berwarna hijau) dan titik pusat (berwarna merah). Deteksi berhasil mengidentifikasi lingkaran dengan akurasi yang baik, baik untuk ukuran besar maupun kecil. Lingkaran yang terdeteksi berada dalam batas radius yang telah ditentukan (minRadius=50, maxRadius=150).

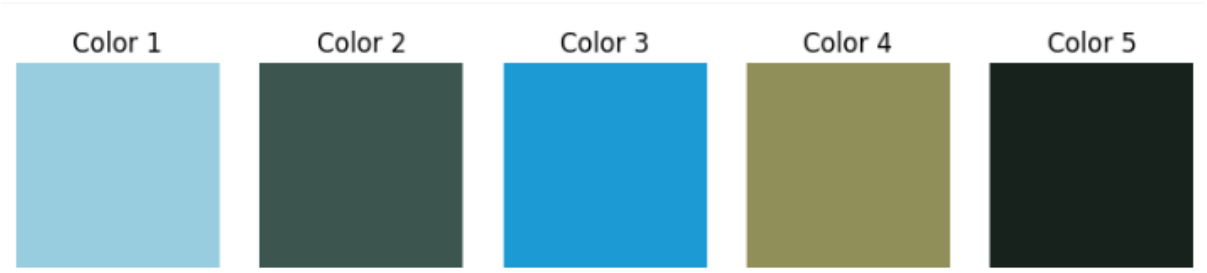
E. Ekstrasi Warna Dominan pada Gambar

Ekstraksi warna dominan dilakukan dengan algoritma K-Means Clustering, yang bertujuan untuk mengelompokkan piksel pada gambar ke dalam beberapa kluster warna dominan. Adapun langkah-langkahnya meliputi:

1. Konversi Gambar: Gambar diubah ke format RGB karena OpenCV membaca gambar dalam format BGR.
2. Reshape Gambar: Matriks gambar 3D diubah menjadi array 2D agar setiap piksel dapat diwakili oleh nilai warna (R, G, B).
3. Clustering: Algoritma K-Means Clustering diterapkan untuk mengelompokkan piksel menjadi lima kluster (k = 5).
4. Warna Dominan: Warna dari setiap kluster direpresentasikan sebagai nilai pusat kluster (centroid).

Berdasarkan langkah-langkah tersebut, dilakukan percobaan terhadap gambar berikut:

Diperoleh hasil sebagai berikut:



Hasil Analisis:

Gambar menghasilkan lima warna utama yang mewakili elemen dominan dalam gambar, yaitu:

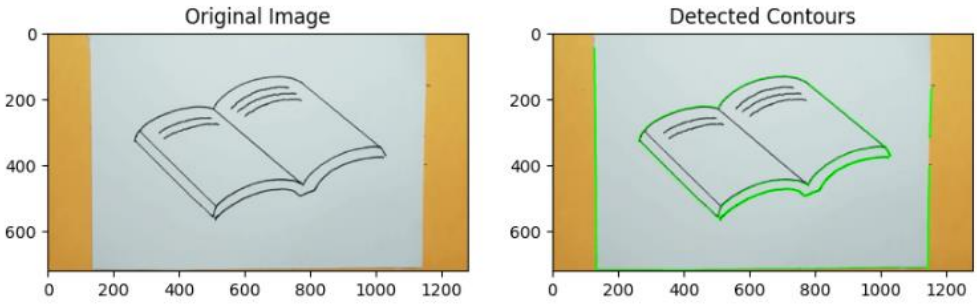
- Color 1: Biru terang, mewakili warna langit.
- Color 2: Hijau gelap, merepresentasikan vegetasi atau dedaunan.
- Color 3: Biru cerah, juga bagian dari langit yang lebih pekat.
- Color 4: Warna hijau-kuning, mencerminkan gunung dan pepohonan.
- Color 5: Hijau tua atau hitam gelap, bagian dari bayangan atau area gelap pada vegetasi.

F. Deteksi Kontur pada Gambar

Langkah-langkah pada percobaan ini meliputi:

1. Konversi Gambar ke Grayscale:
   Gambar asli diubah menjadi skala abu-abu untuk menyederhanakan data warna dan fokus pada intensitas piksel.
2. Gaussian Blur:
   Gaussian blur dengan kernel 7x7 diterapkan untuk mengurangi noise, sehingga deteksi tepi menjadi lebih stabil.
3. Adaptive Threshold:
   Teknik ini digunakan untuk mengatasi variasi pencahayaan dalam gambar, menghasilkan gambar biner di mana objek utama terpisah dari latar belakang.
4. Deteksi Tepi dengan Canny Edge Detection:
   Proses ini mendeteksi tepi dengan mencari area dengan perubahan intensitas yang signifikan.
5. Ekstraksi Kontur:
   Kontur dideteksi menggunakan fungsi cv2.findContours, dan hanya kontur dengan area lebih dari 500 piksel dipilih untuk fokus pada objek besar.
6. Visualisasi Kontur:
   Kontur yang terdeteksi digambar pada gambar asli dengan warna hijau untuk membantu identifikasi.

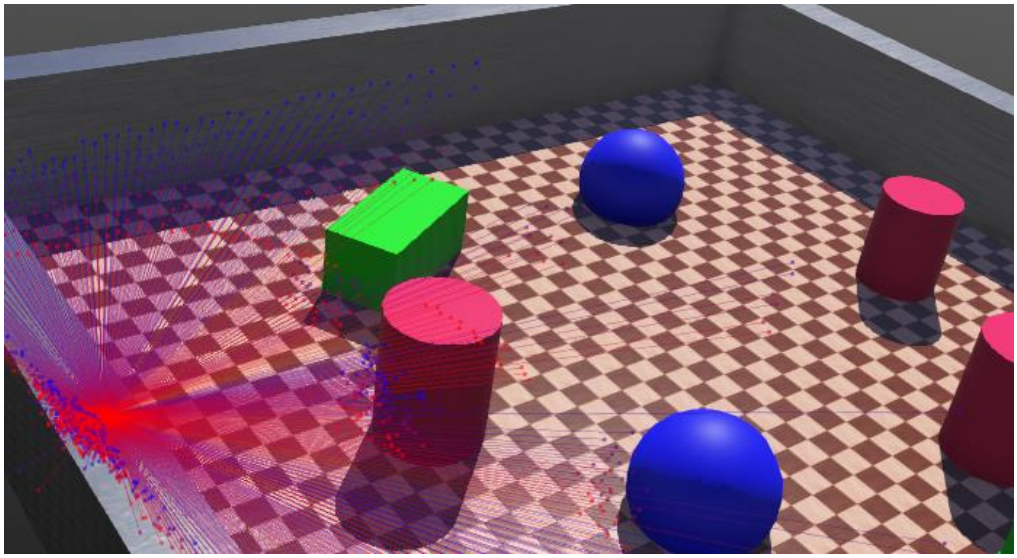
Didapatkan hasil percobaannya yaitu:



Hasil Analisis:

- Gambar Asli: Gambar menunjukkan objek utama (ilustrasi buku) yang terletak di latar belakang putih dengan elemen tambahan (tepi frame berwarna kuning).
- Gambar dengan Kontur: Kontur utama dari buku terdeteksi dengan baik, termasuk detail seperti garis ilustrasi pada halaman. Kontur besar di sekitar frame latar belakang juga terdeteksi.

G. Lidar Data Extraction and Obstacle Detection pada Webots



Sensor LiDAR digunakan untuk mendapatkan data jarak secara menyeluruh dengan cakupan 360 derajat, sementara sensor ultrasonik digunakan untuk mendeteksi rintangan di sisi kiri dan kanan robot pada jarak yang lebih dekat. Program dimulai dengan inisialisasi robot menggunakan API Webots. Sensor LiDAR diaktifkan menggunakan metode lidar.enable() untuk membaca data jarak dan lidar.enablePointCloud() untuk memvisualisasikan data dalam bentuk 3D. Selain itu, dua sensor ultrasonik diaktifkan untuk memberikan informasi tentang rintangan yang berada di dekat sisi kiri dan kanan robot.

Fungsi extract_lidar_data() bertanggung jawab membaca data jarak dari sensor LiDAR dalam bentuk array. Data ini mencakup jarak setiap sudut pandang dalam meter. Nilai yang rendah menunjukkan keberadaan objek yang dekat, sementara nilai inf menunjukkan bahwa tidak ada objek yang terdeteksi dalam jarak jangkauan pada sudut tersebut. Fungsi read_distance_sensors() membaca data jarak dari sensor ultrasonik. Data ini digunakan untuk mendeteksi keberadaan rintangan secara lokal

pada kedua sisi robot. Informasi dari sensor ini memberikan nilai yang lebih kecil ketika rintangan berada sangat dekat dengan robot.

Kecepatan roda robot dihitung menggunakan fungsi compute_speeds(). Fungsi ini mengkombinasikan data dari sensor ultrasonik dengan koefisien empiris untuk menghasilkan kecepatan yang berbeda bagi roda kiri dan kanan. Hal ini memungkinkan robot untuk menghindari rintangan dengan cara membelokkan arah geraknya. Dalam setiap langkah simulasi, robot membaca data dari LiDAR dan sensor ultrasonik, lalu menghitung kecepatan roda kiri dan kanan berdasarkan informasi tersebut. Robot secara otomatis menghindari rintangan dengan mengatur kecepatan roda secara dinamis. Kecepatan dasar yang digunakan adalah 6.0, yang kemudian disesuaikan dengan data sensor.

Hasil simulasi menunjukkan bahwa robot dapat mendeteksi objek-objek di sekitarnya menggunakan LiDAR. Objek yang terdeteksi divisualisasikan dalam bentuk titik-titik (point cloud) berwarna merah dan biru, sementara sensor ultrasonik mendeteksi keberadaan rintangan di area dekat robot. Robot secara efektif mengubah arah geraknya berdasarkan data yang diterima untuk menghindari tabrakan. Program ini berhasil menunjukkan kemampuan navigasi dinamis dengan memanfaatkan data sensor. Sistem ini dapat dikembangkan lebih lanjut dengan mengoptimalkan data LiDAR, seperti menggunakan filter untuk menghilangkan kebisingan, membagi zona deteksi untuk respons yang lebih terarah, dan membangun peta 2D atau 3D dari lingkungan robot. Dengan integrasi teknologi tambahan seperti kamera, sistem ini dapat digunakan untuk aplikasi navigasi otonom yang lebih kompleks.