

Laporan Praktikum 6 Kontrol Cerdas

Nama : Atniko Dwi Saputra

NIM : 224308005

Kelas : TKA – 6A

Akun Github (Tautan) : <https://github.com/atnikodwi>

Student Lab Assistant : Muhammad Mahirul Faiq / 214308043

1. Judul Percobaan

Judul : Canny Edge Detection & Lane Detection with Instance Segmentation

2. Tujuan Percobaan

Pada praktikum ini, mahasiswa akan dapat:

- Memahami konsep **Canny Edge Detection** sebagai metode dasar deteksi tepi.
- Menggunakan **Instance Segmentation** untuk deteksi jalur rel kereta (Lane Detection).
- Menggunakan dataset **Rail Segmentation** dari Kaggle untuk eksperimen.
- Menggabungkan metode **Canny Edge Detection** dengan Instance Segmentation untuk meningkatkan deteksi jalur.

3. Landasan Teori

1. Canny Edge Detection

Canny Edge Detection adalah salah satu metode deteksi tepi yang banyak digunakan dalam pemrosesan citra digital. Algoritma ini dikembangkan oleh John F. Canny pada tahun 1986 dan dirancang untuk mengidentifikasi tepi dalam suatu gambar dengan mengurangi jumlah data yang perlu diproses sekaligus mempertahankan struktur utama dari objek dalam gambar.

Proses utama dalam algoritma Canny Edge Detection meliputi beberapa tahap:

1. Gaussian Filtering: Menghaluskan gambar dengan filter Gaussian untuk mengurangi noise.
2. Gradient Computation: Menggunakan operator Sobel untuk menghitung gradien intensitas dalam arah horizontal dan vertikal.
3. Non-Maximum Suppression: Menghilangkan piksel yang bukan merupakan bagian dari tepi utama dengan membandingkan nilai gradiennya.
4. Double Thresholding: Menentukan batas ambang tinggi dan rendah untuk mengklasifikasikan piksel sebagai tepi kuat atau lemah.
5. Edge Tracking by Hysteresis: Menghubungkan tepi lemah dengan tepi kuat jika berada dalam jarak yang sesuai untuk memastikan tepi yang berkesinambungan.

Metode ini banyak digunakan dalam berbagai aplikasi, termasuk deteksi objek, pengenalan pola, dan pengolahan gambar medis karena kemampuannya dalam menangkap detail yang signifikan dalam gambar.

2. Lane Detection dengan Instance Segmentation

Lane Detection adalah teknik dalam pengolahan citra digital yang digunakan untuk mendeteksi jalur atau garis jalan dalam gambar atau video, biasanya digunakan dalam sistem kendaraan otonom dan asisten pengemudi. Salah satu pendekatan modern dalam Lane Detection adalah dengan menggunakan metode Instance Segmentation.

Instance Segmentation adalah teknik yang tidak hanya membedakan antara berbagai kelas objek dalam gambar tetapi juga mengidentifikasi setiap instance individu dari kelas yang sama. Dalam konteks Lane Detection, metode ini digunakan untuk mendeteksi dan membedakan setiap jalur secara individual.

Proses Lane Detection dengan Instance Segmentation meliputi:

1. Preprocessing: Menghilangkan noise dan meningkatkan kontras gambar agar jalur lebih mudah diidentifikasi.
2. Feature Extraction: Menggunakan model deep learning seperti Convolutional Neural Networks (CNN) atau Mask R-CNN untuk mengenali fitur dari jalur jalan.
3. Segmentation: Menerapkan model segmentasi yang memisahkan jalur dari bagian lain dalam gambar.
4. Post-processing: Menggunakan teknik filtering dan curve fitting untuk mendapatkan jalur yang lebih akurat dan mulus.

Pendekatan ini lebih akurat dibandingkan metode klasik karena mampu menangani kondisi lingkungan yang kompleks, seperti perubahan pencahayaan, kondisi jalan yang tidak teratur, serta berbagai variasi jalur yang ada di jalan raya.

4. Analisis dan Diskusi

Analisis :

1. Seberapa baik deteksi jalur dengan Instance Segmentation dibandingkan Canny?

Perbandingan Deteksi Jalur dengan Instance Segmentation dan Canny Edge Detection.

Instance Segmentation lebih unggul dibandingkan Canny Edge Detection dalam mendeteksi jalur secara akurat karena mampu mengenali jalur secara individu serta mengatasi noise dan variasi warna pada gambar. Sementara itu, Canny Edge Detection lebih sederhana dan cepat tetapi kurang efektif dalam kondisi pencahayaan yang buruk atau latar belakang yang kompleks.

2. Apakah kombinasi kedua metode dapat meningkatkan akurasi?

Kombinasi Canny Edge Detection dan Instance Segmentation.

Menggabungkan kedua metode ini dapat meningkatkan akurasi deteksi jalur. Canny Edge Detection dapat digunakan sebagai langkah awal untuk menyoroti batas jalur sebelum diterapkan Instance Segmentation, sehingga membantu memperjelas struktur jalur dan meningkatkan performa model deteksi.

3. Apa dampak perubahan parameter Canny (thresholds) terhadap hasil deteksi?

Dampak Perubahan Parameter Canny (Thresholds)

Perubahan parameter threshold pada Canny Edge Detection mempengaruhi sensitivitas terhadap tepi. Jika threshold terlalu rendah, terlalu banyak tepi yang terdeteksi, termasuk noise yang tidak diinginkan. Sebaliknya, jika threshold terlalu tinggi, beberapa tepi penting dapat terlewatkan.

Diskusi :

1. Kapan lebih baik menggunakan Canny Edge Detection dibanding Instance Segmentation?

Canny Edge Detection lebih baik digunakan untuk tugas yang membutuhkan pemrosesan cepat dengan sumber daya komputasi terbatas, seperti deteksi objek sederhana atau analisis awal sebelum segmentasi lebih lanjut. Instance Segmentation lebih cocok untuk sistem yang membutuhkan akurasi tinggi, seperti kendaraan otonom.

2. Bagaimana cara meningkatkan deteksi jalur dengan tuning parameter YOLOv8-seg?

Untuk meningkatkan deteksi jalur dengan YOLOv8-seg, beberapa parameter yang dapat disesuaikan meliputi:

- Confidence Threshold: Meningkatkan atau menurunkan batas kepercayaan untuk mengurangi false positives dan meningkatkan deteksi objek yang diinginkan.
- IoU Threshold: Menyesuaikan batas Intersection over Union agar lebih akurat dalam menentukan batas jalur.
- Augmentasi Data: Melatih model dengan variasi gambar seperti perubahan pencahayaan dan rotasi untuk meningkatkan generalisasi model.

3. Bagaimana metode ini dapat diterapkan dalam sistem navigasi kereta otomatis?

Metode deteksi jalur dapat diterapkan dalam sistem navigasi kereta otomatis untuk memastikan kereta tetap berada di jalurnya. Dengan Instance Segmentation, sistem dapat mengenali dan menyesuaikan jalur berdasarkan kondisi lingkungan secara real-time, sementara Canny Edge Detection dapat digunakan sebagai validasi tambahan untuk meningkatkan keandalan deteksi.

5. Assignment

Program ini memanfaatkan MediaPipe dan OpenCV untuk mendeteksi serta melacak wajah dan Praktikum keenam ini bertujuan untuk mengembangkan sistem yang mampu mendeteksi tepi menggunakan metode Canny Edge Detection serta mengombinasikannya dengan deteksi objek atau segmentasi menggunakan model YOLO dari Ultralytics. Sistem ini bekerja dengan menangkap video secara real-time melalui kamera.

Proses dimulai dengan memuat model YOLO dari lokasi yang telah ditentukan menggunakan perintah YOLO(MODEL_PATH), memastikan bahwa model tersedia sebelum memproses video dengan OpenCV. Kamera diakses melalui cv2.VideoCapture(0), lalu setiap frame yang diambil dikonversi ke skala abu-abu menggunakan cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) agar dapat dilakukan deteksi tepi dengan metode Canny Edge Detection. Proses ini menghasilkan gambar biner dengan tepi putih di atas latar belakang hitam melalui cv2.Canny(gray, low_threshold, high_threshold).

Sementara itu, frame asli dikonversi ke format RGB menggunakan cv2.cvtColor(frame, cv2.COLOR_BGR2RGB), kemudian diproses oleh model YOLO untuk mendeteksi objek atau melakukan segmentasi. Hasil prediksi pertama disimpan dalam variabel predictions = results[0]. Jika model menggunakan segmentasi, area objek yang terdeteksi ditandai dengan poligon biru transparan menggunakan cv2.fillPoly(overlay, [mask], (255, 0, 0)). Jika model hanya mendeteksi objek dalam bentuk bounding box, objek tersebut akan dikelilingi kotak hijau dengan cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2), dan label objek akan ditampilkan jika tersedia menggunakan cv2.putText().

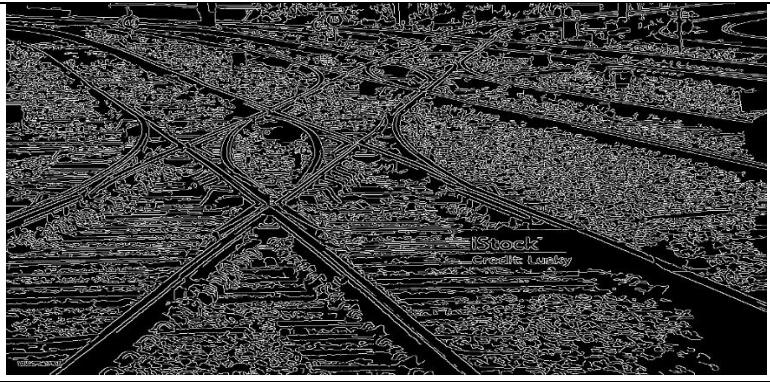
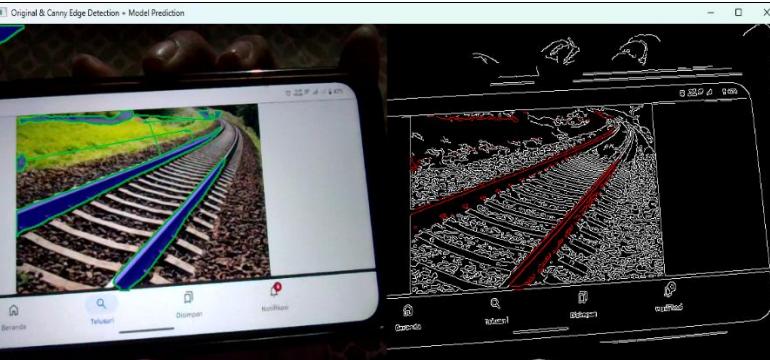
Untuk membedakan tepi objek yang dikenali YOLO dengan hasil deteksi tepi dari metode Canny, digunakan masking. Tepi dalam area objek yang terdeteksi YOLO akan diberi warna merah, sementara bagian lainnya tetap berwarna putih dengan edges_bgr[np.where((mask_model == 255) & (edges == 255))] = [0, 0, 255]. Hasil akhir berupa frame yang mengandung anotasi YOLO serta

hasil deteksi tepi ditampilkan berdampingan menggunakan np.hstack(), sehingga perbedaan antara tepi umum dan tepi objek lebih terlihat jelas.

Program berjalan dalam loop hingga pengguna menekan tombol 'q', yang akan memicu perintah cv2.waitKey(1) & 0xFF == ord('q') untuk keluar. Setelah itu, kamera dilepaskan menggunakan cap.release() dan semua jendela yang terbuka ditutup dengan cv2.destroyAllWindows() guna mencegah kebocoran memori. Dengan menggabungkan metode deteksi tepi tradisional dan deep learning untuk segmentasi atau deteksi objek, sistem ini dapat diterapkan dalam berbagai bidang seperti deteksi rel kereta, pengawasan lalu lintas, dan aplikasi lainnya.

6. Data dan Output Hasil Pengamatan

Sajikan data dan hasil yang diperoleh selama percobaan. Gunakan tabel untuk menyajikan data jika diperlukan.

No	Variabel	Hasil Pengamatan
1	Canny edge	
2	Lane detection	
3	Combined detection	

7. Kesimpulan

Canny Edge Detection merupakan metode deteksi tepi yang sederhana dan cepat, tetapi kurang akurat dalam kondisi pencahayaan yang buruk. Sementara itu, Instance Segmentation dengan YOLOv8-seg lebih unggul dalam mendeteksi jalur karena mampu mengenali setiap jalur secara individual dan menangani variasi lingkungan yang kompleks. Kombinasi kedua metode memberikan hasil yang lebih baik dengan memanfaatkan keunggulan masing-masing metode,

yaitu kecepatan Canny Edge Detection dan akurasi Instance Segmentation. Tuning parameter pada YOLOv8-seg dapat meningkatkan kinerja deteksi jalur dengan menyesuaikan threshold confidence dan IoU serta melakukan augmentasi data. Metode ini dapat diterapkan pada sistem navigasi kereta otomatis untuk memastikan kereta tetap berada di jalurnya.

8. Saran

Untuk meningkatkan akurasi deteksi jalur, sebaiknya dilakukan pengolahan data lebih lanjut seperti filtering dan edge refinement sebelum tahap segmentasi. Tuning parameter YOLOv8-seg secara optimal juga dapat meningkatkan akurasi deteksi dan mengurangi kesalahan dalam mendeteksi jalur. Penggunaan dataset yang lebih variatif dengan berbagai kondisi pencahayaan dan lingkungan akan meningkatkan kemampuan model dalam mendeteksi jalur secara lebih akurat. Selain itu, eksplorasi model deep learning lainnya, seperti Mask R-CNN atau DeepLabV3, dapat dilakukan untuk mendapatkan model yang lebih optimal untuk deteksi jalur.

9. Daftar Pustaka

- Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679-698.
- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961-296