BAB II

TINJAUAN LITERATUR

2.1 Penelitian Terkait

Penelitian terkait merupakan tinjauan terhadap penelitian-penelitian sebelumnya yang memiliki hubungan atau relevansi dengan topik penelitian yang sedang dilakukan yaitu mengenai deteksi lubang jalan. Penelitian terkait ini digunakan untuk mengetahui perkembangan penelitian terdahulu, metode yang telah digunakan, serta hasil yang telah diperoleh, yang dapat menjadi dasar dilakukannya penelitian baru. Berikut beberapa penelitian terkait deteksi lubang pada jalan melalui pengolahan citra.

Penelitian ini menggunakan dataset sebanyak 2952 citra lubang jalan yang diperoleh dari Roboflow, kemudian dilakukan proses pelatihan model dengan pembagian data menjadi training, validation, dan testing serta penerapan data augmentation untuk meningkatkan variasi data. Hasil penelitian menunjukkan bahwa model mampu mendeteksi lubang jalan dengan performa yang cukup baik dengan nilai mAP@0.5 sebesar 0,8, precision 75,1%, recall 76,6%, dan F1-score 75,8%. Sistem juga diimplementasikan dalam bentuk aplikasi berbasis web menggunakan Flask yang memungkinkan pengguna mengunggah citra jalan untuk dideteksi secara otomatis sekaligus menghitung estimasi luas lubang menggunakan metode kontur dari OpenCV. Hasil penelitian ini menunjukkan bahwa pendekatan YOLOv11 efektif digunakan untuk mendeteksi lubang jalan secara otomatis guna

mendukung pemantauan dan pemeliharaan infrastruktur jalan secara lebih efisien (Takyudin *et al.*, 2025).

Penelitian ini menggunakan 200 citra jalan yang telah dianotasi secara manual dengan dua kategori kerusakan utama yaitu retak dan lubang jalan, kemudian dibagi menjadi data training, validation, dan testing dengan rasio 70:20:10. Model dilatih selama 100 epoch dan dievaluasi menggunakan metrik precision, recall, dan mean Average Precision (mAP). Hasil penelitian menunjukkan bahwa model mampu mendeteksi kerusakan jalan dengan performa yang cukup baik, dengan nilai precision mencapai sekitar 0,4, recall lebih dari 0,3, serta mAP50 sekitar 0,27. Hasil tersebut menunjukkan bahwa algoritma YOLOv8 memiliki potensi untuk digunakan dalam sistem deteksi kerusakan jalan secara otomatis dan real-time guna mendukung pemantauan kondisi infrastruktur jalan secara lebih efisien (Sofiani, Nazwa Eka Hervy, 2025).

Mengimplementasikan metode deep learning menggunakan algoritma YOLOv8 untuk mendeteksi kerusakan jalan secara otomatis dan real-time. Dataset yang digunakan adalah RDD2022 (Road Damage Dataset 2022) yang berjumlah 26.620 citra dengan anotasi berbagai jenis kerusakan jalan seperti retak dan lubang. Proses penelitian meliputi tahap preprocessing, augmentasi data, pelatihan model, dan evaluasi performa menggunakan metrik precision, recall, F1-score, dan mean Average Precision (mAP). Hasil pengujian menunjukkan bahwa model YOLOv8 mampu mencapai akurasi deteksi sebesar 86,4%, precision 67,3%, recall 65,2%, F1-score 0,61, serta mAP@0.5 sebesar 62% dengan kecepatan inferensi sekitar 0,5 ms, sehingga menunjukkan bahwa metode ini cukup efektif dan efisien untuk

diterapkan pada sistem deteksi kerusakan jalan secara real-time (Agus Prastyo, 2025).

Mengembangkan sistem deteksi kerusakan jalan menggunakan metode deep learning berbasis YOLOv5 untuk mengidentifikasi berbagai jenis kerusakan seperti retakan dan lubang jalan pada citra permukaan jalan. Dataset yang digunakan berupa kumpulan citra jalan yang telah dianotasi, kemudian dilakukan proses pelatihan model, validasi, dan pengujian untuk mengevaluasi kinerja sistem. Berdasarkan hasil pengujian, model yang dikembangkan mampu mendeteksi kerusakan jalan dengan nilai mAP sebesar 0,89, precision sebesar 0,86, dan recall sebesar 0,83, yang menunjukkan bahwa metode YOLOv5 memiliki performa yang cukup baik dalam mengenali objek kerusakan jalan pada citra digital. Hasil penelitian ini menunjukkan bahwa pendekatan deep learning dapat digunakan sebagai solusi yang efektif untuk membantu proses pemantauan kondisi jalan secara otomatis dan lebih efisien dibandingkan metode manual (Putri, 2024).

Mengembangkan sistem deteksi lubang jalan (pothole) menggunakan metode Convolutional Neural Network (CNN) dengan arsitektur SqueezeNet untuk mendukung deteksi secara real-time menggunakan kamera kendaraan. Penelitian ini membandingkan beberapa arsitektur CNN yaitu SqueezeNet, ResNet152, dan InceptionNet untuk mengetahui performa terbaik dalam mendeteksi lubang jalan pada citra. Evaluasi kinerja model dilakukan menggunakan metrik precision, recall, F1-score, dan accuracy. Hasil penelitian menunjukkan bahwa InceptionNet memperoleh akurasi tertinggi sebesar 93,3%, diikuti ResNet152 dengan akurasi 85,3%, sedangkan SqueezeNet memperoleh akurasi 79,3%, namun memiliki

keunggulan pada ukuran model yang lebih kecil dan kecepatan proses yang lebih tinggi, sehingga lebih sesuai untuk implementasi pada sistem deteksi lubang jalan secara real-time (Lehman *et al.*, 2025).

Penelitian ini dilatarbelakangi oleh tingginya angka kecelakaan lalu lintas akibat jalan berlubang, sehingga diperlukan sistem otomatis yang mampu mendeteksi dan membedakan lubang dari objek lain seperti bayangan. Dataset yang digunakan terdiri dari 11.196 citra yang telah melalui tahap prapemrosesan berupa cropping, konversi grayscale, dan resizing ke ukuran 128x64 piksel. PHOG diterapkan dengan parameter 5 level dan 9 bins orientasi, menghasilkan vektor fitur berdimensi 12.285. Hasil pengujian menunjukkan bahwa kinerja terbaik diperoleh pada penggunaan PHOG level 4, smoothing sebagai image enhancement, dan kernel polynomial pada SVM dengan parameter C=10 dan gamma=0.001, yang menghasilkan akurasi sebesar 94,45%, presisi 96,13%, recall 95,77%, dan F1-score 95,95%. Penelitian ini membuktikan bahwa PHOG lebih unggul dibandingkan HOG single level dalam merepresentasikan ciri bentuk jalan berlubang, sehingga sistem yang dibangun mampu mengenali jalan berlubang dengan sangat baik (Fitriansyah, Rachmawati and Risnandar, 2023).

Penelitian ini dilatarbelakangi oleh tingginya risiko kecelakaan dan biaya perawatan kendaraan akibat jalan berlubang, serta keterbatasan metode manual yang membutuhkan waktu dan tenaga besar. Sistem dirancang menggunakan metodologi Agile dengan perangkat keras berupa ESP32-CAM yang terhubung ke power bank dan komputer untuk pemrosesan data. Dataset dilatih menggunakan YOLOv8 di Google Colab menghasilkan precision 0,70, recall 0,78, dan F1-

Score 0,73. Faktor-faktor yang memengaruhi kinerja sistem meliputi kondisi pencahayaan, kompleksitas objek (bayangan yang mirip lubang), kapasitas daya, serta keterbatasan variasi dataset. Penelitian ini membuktikan bahwa integrasi ESP32-CAM dengan YOLOv8 layak dikembangkan sebagai sistem deteksi lubang jalan real-time, meskipun masih memerlukan komputer pendukung dan memiliki ruang peningkatan pada akurasi serta kemandirian system (Royhan *et al.*, 2025).

Penelitian ini dilatarbelakangi oleh keterbatasan metode inspeksi visual konvensional yang membutuhkan banyak waktu, tenaga ahli, dan biaya besar, serta tidak efektif untuk cakupan jalan yang luas. Sistem dirancang untuk mendeteksi enam jenis kerusakan jalan yaitu retakan, block, longitudinal, transverse, lubang, dan tambalan. Arsitektur YOLOv8 yang digunakan memiliki jaringan dengan konvolusi dan berbasis ResNet yang menyederhanakan proses prediksi. Hasil penelitian menunjukkan bahwa model YOLOv8 mampu mendeteksi kerusakan jalan dengan sangat baik, yang ditunjukkan dengan nilai F1-Score mencapai 0,99 pada confidence threshold 0,822, precision mencapai 1,0 pada ambang keyakinan 0,852, serta confusion matrix yang menunjukkan 39 prediksi benar untuk kelas rusak dan hanya satu false positive (latar belakang terdeteksi sebagai kerusakan). Penelitian ini membuktikan bahwa pendekatan transfer learning dengan YOLOv8 efektif untuk deteksi kerusakan jalan di Indonesia, meskipun masih memerlukan penambahan data lokal yang lebih seimbang untuk meningkatkan performa model (Denaro and Lim, 2025).

Penelitian ini bertujuan mengembangkan sistem deteksi kerusakan jalan menggunakan teknologi deep learning dan pengolahan citra, dengan memanfaatkan

algoritma YOLOv4-Tiny yang dikombinasikan dengan data posisi dari GNSS untuk memberikan lokasi akurat pada setiap kerusakan terdeteksi. Model dilatih menggunakan dataset GRDDC dengan kategori kerusakan seperti retak linier, retak buaya, lubang, dan marka jalan pudar. Hasil pengujian menunjukkan model memiliki mean Average Precision (mAP) sebesar 41%, average loss 0,6428, serta akurasi overall sebesar 88% dan akurasi kappa 86% berdasarkan analisis confusion matrix terhadap 85 sampel validasi. Sistem ini diharapkan dapat menggantikan metode survei manual yang memakan waktu dan sumber daya, serta mendukung pemeliharaan jalan secara lebih cepat dan tepat sasaran (Sasmito, Setiadji and Isnanto, 2023).

Penelitian ini bertujuan mengembangkan sistem deteksi kerusakan jalan berdasarkan citra digital menggunakan metode Convolutional Neural Network (CNN) untuk mengklasifikasikan kondisi jalan menjadi "Rusak" atau "Tidak Rusak" guna membantu instansi terkait dalam penanganan cepat terhadap infrastruktur jalan yang rusak. Dataset diperoleh dari situs kaggle.com sebanyak 200 citra digital yang dikategorikan ke dalam empat kelas yaitu Berlubang, Retak, Lumpur, dan Bebatuan, melalui tahapan preprocessing seperti resize dan crop gambar serta pembagian data dengan rasio 50:30:20 dan 60:20:20. Berdasarkan percobaan model dengan berbagai konfigurasi epochs (20 hingga 100), model terbaik diperoleh pada percobaan ke-3 dengan rasio 70:30 dan 80 epochs yang menghasilkan akurasi training 92% dan akurasi validasi 94%, dengan nilai loss masing-masing 0,16 dan 0,10. Pengujian model menggunakan confusion matrix terhadap 80 gambar uji menunjukkan akurasi sebesar 82%, presisi 85%, recall 83%,

dan f1-score 82%. Hasil ini mengindikasikan bahwa model CNN yang dikembangkan memiliki kinerja stabil dan layak digunakan untuk deteksi kerusakan jalan, meskipun terdapat potensi overfitting yang perlu diantisipasi dengan pengaturan parameter lebih lanjut dan penambahan variasi data (Mulyana and Wahyudi, 2025).

Tabel 2.1 Pustaka

No.	Penelitian (Tahun)	Metode / Algoritma	Dataset & Ukuran Data	Hasil / Metrik Performa
1	Takyudin et al. (2025)	YOLOv11, data augmentation, aplikasi web berbasis Flask, estimasi luas dengan kontur OpenCV	2.952 citra lubang jalan (Roboflow)	mAP@0.5 = 0,80; Precision = 75,1%; Recall = 76,6%; F1-Score = 75,8%
2	Sofiani, Nazwa Eka Hervy (2025)	YOLOv8, anotasi manual (retak & lubang), 100 epoch	200 citra (rasio 70:20:10)	Precision ≈ 0,4; Recall > 0,3; mAP50 ≈ 0,27
3	Agus Prastyo (2025)	YOLOv8, preprocessing & augmentasi data	RDD2022 – 26.620 citra	Akurasi = 86,4%; Precision = 67,3%; Recall = 65,2%; F1-Score = 0,61; mAP@0.5 = 62%; inferensi ≈ 0,5 ms
4	Putri (2024)	YOLOv5	Citra jalan teranotasi (retak & lubang)	mAP = 0,89; Precision = 0,86; Recall = 0,83
5	Lehman et al. (2025)	CNN — perbandingan SqueezeNet, ResNet152, InceptionNet	Citra lubang jalan (kamera kendaraan)	Akurasi tertinggi: InceptionNet 93,3%; ResNet152 85,3%; SqueezeNet 79,3% (ukuran model lebih kecil & lebih cepat)
6	Fitriansyah, Rachmawati & Risnandar (2023)	PHOG (5 level, 9 bins) + SVM kernel polynomial (C=10, gamma=0,001), preprocessing grayscale & resize 128×64	11.196 citra	Akurasi = 94,45%; Presisi = 96,13%; Recall = 95,77%; F1-Score = 95,95%
7	Royhan et al. (2025)	YOLOv8 + hardware ESP32-CAM, metodologi	Dataset lubang jalan (real-time capture)	Precision = 0,70; Recall = 0,78; F1-Score = 0,73

		Agile, training di Google Colab		
8	Denaro & Lim (2025)	YOLOv8 (backbone ResNet), 450 epoch, 6 kelas kerusakan jalan	Dataset multi-kelas kerusakan jalan	F1-Score = 0,99 (threshold 0,822); Precision = 1,0 (threshold 0,852); 39 TP, 1 FP pada confusion matrix
9	Sasmito, Setiadji & Isnanto (2023)	YOLOv4-Tiny + data posisi GNSS	Dataset GRDDC	mAP = 41%; Average loss = 0,6428; Akurasi = 88%; Kappa = 86% (85 sampel validasi)
10	Mulyana & Wahyudi (2025)	CNN, klasifikasi 4 kelas (Berlubang, Retak, Lumpur, Bebatuan), preprocessing resize & crop	200 citra (Kaggle), rasio 70:30, 80 epoch	Akurasi training = 92%; Akurasi validasi = 94%; Akurasi uji = 82%; Presisi = 85%; Recall = 83%; F1-Score = 82%

Berdasarkan kajian pustaka tersebut, dapat disimpulkan bahwa penelitian pengolahan citra digital di Indonesia telah banyak membahas deteksi dan klasifikasi lubang jalan, namun penelitian yang secara spesifik berfokus pada pengukuran luas lubang jalan masih relatif terbatas. Oleh karena itu, penelitian ini diarahkan untuk mengembangkan sistem pengukuran luas lubang jalan berbasis pengolahan citra digital.

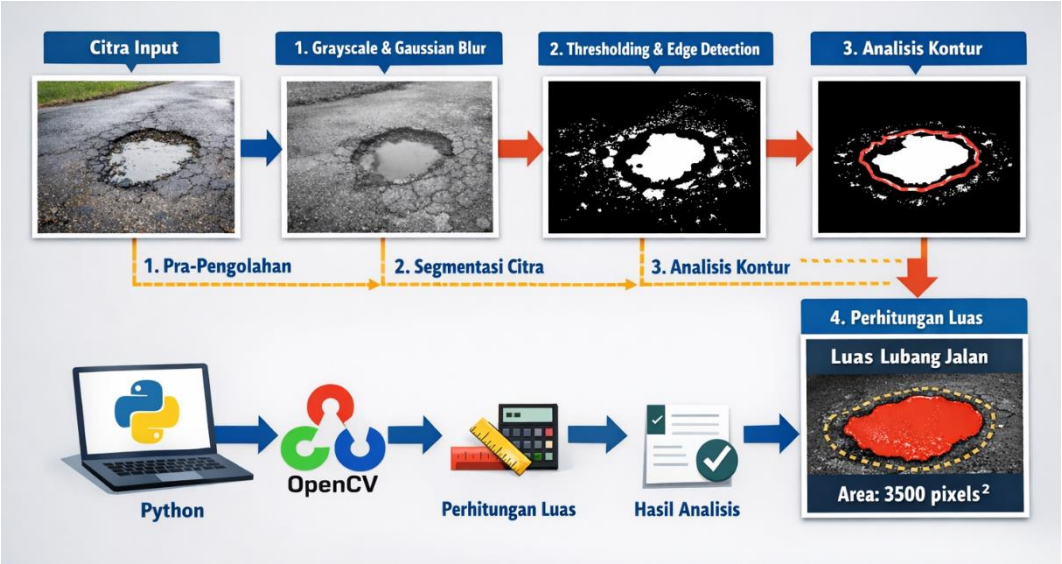
2.2 Pengolahan Citra Digital

Pengolahan citra digital merupakan suatu proses yang bertujuan untuk memanipulasi dan menganalisis citra menggunakan komputer agar menghasilkan citra dengan kualitas lebih baik atau mengekstrak informasi tertentu dari citra tersebut . Citra digital sendiri tersusun oleh unit-unit kecil yang disebut piksel, di mana setiap piksel memiliki nilai intensitas yang menunjukkan tingkat keabuan atau warna . Teknik pengolahan citra dapat diterapkan pada berbagai bidang seperti

penginderaan jauh, robotik, pemetaan, dan biomedis . Ruang lingkup pengolahan citra meliputi perbaikan kualitas citra (image enhancement), restorasi citra (image restoration), segmentasi citra (image segmentation), serta representasi dan deskripsi citra . Melalui berbagai tahapan tersebut, pengolahan citra mampu melakukan analisis seperti perhitungan luas permukaan objek berdasarkan jumlah pikselnya atau ekstraksi fitur geometris lainnya (Irama and Sandy, 2025).

Sistem Pengukuran Luas Lubang Jalan Menggunakan Pengolahan Citra Digital Berbasis OpenCV, konsep ini menjadi fondasi utama dalam perancangan sistem. Tujuan pengolahan citra dalam konteks ini adalah untuk memperoleh informasi kuantitatif berupa luas lubang jalan secara otomatis. Proses diawali dengan akuisisi citra lubang jalan menggunakan kamera , kemudian citra RGB hasil akuisisi diubah menjadi citra grayscale untuk menyederhanakan informasi (Alamsyah and Wijaya, 2025). Tahap selanjutnya adalah segmentasi menggunakan metode thresholding Otsu untuk memisahkan objek lubang dari latar belakang jalan dengan menghasilkan citra biner. Operasi morfologi seperti erosi dan dilasi diterapkan untuk menyempurnakan hasil segmentasi dengan menghilangkan noise (Mar'ah *et al.*, 2025). Setelah objek lubang terisolasi dengan baik, dilakukan perhitungan jumlah piksel yang membentuk area lubang tersebut untuk merepresentasikan luas dalam satuan piksel . Luas dalam piksel ini kemudian dikonversi menjadi satuan metrik (cm^2 atau m^2) dengan memanfaatkan nilai skala yang diperoleh dari data ukuran sebenarnya (panjang dan lebar) lubang di lapangan . Dengan demikian, pengolahan citra digital memungkinkan sistem berbasis OpenCV untuk melakukan pengukuran luas lubang jalan secara otomatis,

cepat, dan objektif, mengatasi keterbatasan metode manual yang selama ini dilakukan menggunakan alat ukur sederhana seperti roll meter dengan bantuan tenaga manusia sepenuhnya.



Gambar 2.1 Alur pemprosesan dalam pengolahan citra (Ghofur *et al.*, 2025)

Gambar ini menunjukkan alur proses pengolahan citra dalam mendeteksi serta menghitung luas lubang jalan menggunakan metode pengolahan citra digital. Proses dimulai dari citra input berupa gambar permukaan jalan yang memiliki lubang, kemudian dilakukan tahap pra-pengolahan dengan mengubah citra menjadi grayscale dan menerapkan Gaussian blur untuk mengurangi noise. Selanjutnya dilakukan segmentasi citra menggunakan teknik thresholding dan deteksi tepi (edge detection) untuk memisahkan objek lubang dari latar belakang. Setelah itu dilakukan analisis kontur untuk mengidentifikasi bentuk lubang jalan. Tahap terakhir adalah perhitungan luas objek berdasarkan area kontur yang terdeteksi sehingga diperoleh estimasi luas lubang jalan dalam satuan piksel. Proses ini

diimplementasikan menggunakan bahasa pemrograman Python dengan pustaka OpenCV untuk mendukung analisis citra secara otomatis.

2.3 Preprocessing

Pre-processing citra merupakan tahapan awal dalam pengolahan citra digital yang bertujuan untuk meningkatkan kualitas citra sebelum dilakukan proses analisis lebih lanjut. Tahap ini sangat penting karena citra yang diperoleh dari kamera atau dataset sering kali memiliki gangguan seperti noise, variasi pencahayaan, atau detail yang tidak relevan dengan objek yang akan dianalisis (Puspita, 2024). Melalui proses pre-processing, citra dapat dipersiapkan sehingga informasi penting pada citra menjadi lebih jelas dan memudahkan proses segmentasi maupun deteksi objek. Secara umum, tahapan pre-processing meliputi grayscale, Gaussian Blur, CLAHE (peningkatan kontras), thresholding, morfologi dan analisis kontur.

2.3.1 Grayscale

Tahap pertama dalam proses pengolahan citra pada sistem ini adalah konversi citra masukan dari format warna (BGR/RGB) menjadi citra grayscale (skala keabuan). Citra warna pada dasarnya tersusun dari tiga kanal warna yang masing-masing memiliki rentang nilai intensitas 0 hingga 255, sehingga pengolahan langsung pada citra warna membutuhkan beban komputasi yang lebih besar (Pramudiya *et al.*, 2024). Dengan mengonversi citra menjadi satu kanal intensitas saja, proses pengolahan pada tahap-tahap selanjutnya seperti penghalusan dan thresholding dapat dilakukan dengan lebih efisien tanpa kehilangan informasi struktural yang dibutuhkan untuk mendeteksi area lubang jalan.

$$Gray(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y)$$

Keterangan:

Gray(x,y)	: Nilai intensitas keabuan pada koordinat piksel (x,y)
R(x,y)	: Nilai kanal warna merah (*Red*) pada piksel (x,y)
G(x,y)	: Nilai kanal warna hijau (*Green*) pada piksel (x,y)
B(x,y)	: Nilai kanal warna biru (*Blue*) pada piksel (x.y)
0,299;0,587;0,114	: koefisien sensitivitas visual manusia terhadap warna

Dalam implementasinya, proses konversi ini dilakukan menggunakan pustaka *OpenCV* melalui fungsi dengan parameter COLOR_BGR2GRAY, yang secara otomatis menerapkan rumus luminositas di atas pada setiap piksel citra input.

2.3.2 Gussian Blur

Setelah citra berhasil diubah menjadi grayscale, tahap selanjutnya adalah penghalusan citra menggunakan filter Gaussian Blur. Filter ini bekerja dengan menerapkan kernel konvolusi berbentuk distribusi normal (Gaussian) ke seluruh piksel citra, sehingga setiap piksel akan dihaluskan berdasarkan rata-rata terbobot dari piksel-piksel tetangganya. Tujuan utama dari penerapan Gaussian Blur adalah untuk mengurangi noise dan detail tekstur kecil yang tidak relevan, sehingga citra menjadi lebih halus dan mudah diproses pada tahap segmentasi (Qonita *et al.*, 2026). Pada sistem ini, ukuran kernel yang digunakan cukup besar, yaitu 11×11, yang dipilih secara khusus untuk meredam tekstur permukaan aspal yang cenderung kasar dan tidak rata. Apabila ukuran kernel terlalu kecil, tekstur aspal yang kasar tersebut berpotensi terdeteksi sebagai area lubang palsu (false positive) pada tahap thresholding, sehingga pemilihan ukuran kernel yang lebih besar menjadi salah satu

bentuk penyesuaian agar sistem lebih tahan terhadap variasi tekstur permukaan jalan di lapangan.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x2+y2}{2\sigma^2}}$$

Keterangan:

G(x,y) : Nilai bobot kernel Gaussian pada koordinat (x, y)relatif terhadap titik pusat kernel

x,y : Jarak horizontal dan vertikal suatu piksel terhadap titik pusat kernel

σ (sigma) : Standar deviasi yang menentukan lebar distribusi/tingkat penyebaran bobot Gaussian

π : Konstanta matematika (≈ 3,14159)

e : Basis logaritma natural (≈ 2,71828)

Gaussian Blur bekerja dengan cara mengonvolusikan (*convolution*) citra dengan sebuah kernel yang nilainya dihasilkan dari fungsi distribusi Gaussian dua dimensi.

2.3.3 CLAHE (Contrast Limited Adaptive Histogram Equalization)

Tahap berikutnya dalam proses pra-pengolahan adalah peningkatan kontras citra menggunakan metode CLAHE (Contrast Limited Adaptive Histogram Equalization). CLAHE merupakan varian dari metode histogram equalization konvensional yang bekerja secara adaptif, yaitu dengan membagi citra ke dalam beberapa blok kecil (tile) dan melakukan perataan histogram pada masing-masing blok secara terpisah, bukan pada keseluruhan citra sekaligus (Attaqwa *et al.*, 2024). Pendekatan ini bertujuan untuk meningkatkan kontras lokal pada area-area tertentu tanpa memperbesar noise secara berlebihan, yang sering menjadi kelemahan utama

metode histogram equalization global. Penggunaan parameter clipLimit berfungsi untuk membatasi tingkat penguatan kontras agar tidak menimbulkan efek noise yang berlebihan, sementara tileGridSize menentukan ukuran blok yang digunakan dalam proses pembagian citra. Pada sistem deteksi lubang jalan ini, CLAHE diterapkan dengan clipLimit=2.0 dan tileGridSize=(8,8), dengan tujuan agar area lubang jalan yang umumnya memiliki intensitas piksel lebih gelap dapat tampak lebih kontras dibandingkan area aspal di sekitarnya, sehingga proses thresholding pada tahap selanjutnya dapat memisahkan kedua area tersebut dengan lebih akurat.

$$h_{clipped}(i) = \begin{cases} \beta, & Jika\ h(i) > \beta \\ h(i), & Jika\ h(i) \leq \beta \end{cases}$$

Keterangan:

ḥ(i) : Nilai histogram asli (jumlah piksel) pada tingkat keabuan i

hclipped(i) : Nilai histogram setelah dibatasi (*clipped*)

β : Nilai *clip limit*, yaitu batas maksimum tinggi histogram yang diizinkan

Piksel-piksel yang terpotong (nilai histogram yang melebihi β) tidak dihilangkan, melainkan didistribusikan kembali (*redistribution*) secara merata ke seluruh tingkat keabuan lainnya dalam tile yang sama, sehingga jumlah total piksel dalam histogram tetap konsisten.

2.3.4 Segmentasi (Adaptive Thrsoholding dan Global Thrsoholding)

Adaptive thresholding adalah salah satu teknik segmentasi citra yang digunakan untuk mengubah citra grayscale menjadi citra biner, dengan cara

menentukan nilai ambang (threshold) secara dinamis dan berbeda-beda untuk setiap wilayah kecil pada citra, alih-alih menggunakan satu nilai ambang tunggal untuk keseluruhan citra. Sedangkan Global thresholding merupakan teknik segmentasi citra yang lebih sederhana, yaitu dengan menentukan satu nilai ambang tunggal yang berlaku seragam untuk seluruh piksel pada citra. Setiap piksel akan dibandingkan dengan nilai ambang tersebut, apabila intensitasnya berada di bawah ambang maka piksel akan diklasifikasikan ke dalam satu kelas (misalnya objek/foreground), sedangkan jika berada di atas ambang akan diklasifikasikan ke kelas lainnya (latar belakang/background). Metode ini bekerja optimal pada citra dengan distribusi intensitas yang jelas dan kontras tinggi antara objek dan latar belakangnya, namun memiliki kelemahan berupa kurang sensitif terhadap variasi pencahayaan yang berbeda-beda di berbagai bagian citra.

$$T(x,y) = \mu\_Gaussian(x,y) - C$$

$$B_{1(x,y)} = \begin{matrix} 255, jika\ I(x,y) < T(x,y) \\ 0, jika\ I(x,y) \geq T(x,y) \end{matrix}$$

Keterangan:

T(x,y) : Nilai ambang (*threshold*) lokal pada koordinat (x, y)

μGaussian(x,y): Rata-rata terboboti Gaussian dari intensitas piksel di sekitar (x, y), dalam jendela berukuran *block_size*

C : Konstanta yang dikurangkan dari rata-rata lokal, berfungsi menggeser nilai ambang

I(x,y) : Nilai intensitas piksel citra hasil CLAHE pada koordinat $(x'y)$

B1(x,y) : Nilai piksel biner hasil adaptive thresholding (255 =

objek/luban, 0 = latar)

Pada sistem ini digunakan ukuran blok (*block_size*) = **31×31** piksel dan konstanta C= **8**, sehingga piksel yang nilainya lebih gelap dari rata-rata lokal di sekitarnya (dikurangi 8) akan dianggap sebagai bagian dari lubang jalan.

Selain *adaptive thresholding*, digunakan pula *thresholding* biner sederhana dengan nilai ambang tetap (global) untuk menangkap area yang secara absolut sangat gelap, dirumuskan sebagai:

$$B_{2(x,y)} = \begin{matrix} 255, jika\ I(x,y) < T_v \\ 0, jika\ I(x,y) \geq T_v \end{matrix}$$

Keterangan:

B2(x,y) : Nilai piksel biner hasil *global thresholding*

T_v : Nilai ambang tetap (*threshold value*)

2.3.5 Operasi Morfologi (Opening dan Closing)

Operasi morfologi merupakan teknik pengolahan citra yang digunakan untuk memanipulasi struktur atau bentuk objek pada citra biner (hasil thresholding), dengan tujuan memperbaiki kualitas hasil segmentasi, baik dengan menghilangkan noise yang tidak diinginkan maupun menyempurnakan bentuk objek yang seharusnya menjadi satu kesatuan. Operasi ini bekerja dengan menggeser sebuah structuring element (kernel) ke seluruh bagian citra, kemudian membandingkan bentuk kernel tersebut dengan piksel-piksel di sekitarnya untuk menentukan apakah suatu piksel dipertahankan, dihilangkan, atau ditambahkan. Pada penelitian ini, digunakan dua jenis operasi morfologi dasar, yaitu opening dan closing, yang masing-masing dibangun dari kombinasi dua operasi fundamental erosi dan dilasi.

$$B_{open} = B \circ K_{open} = (B \ominus K_{open}) \oplus K_{open}$$

Keterangan:

B_{open} : Citra biner hasil oprasi opening

K_{open} : Elemen struktur untuk oprasi opening, berbentuk elips *(ellipse)* beukuran 5x5

Operasi opening bertujuan untuk menghilangkan derau berupa titik-titik atau bintik-bintik kecil yang muncul akibat tekstur permukaan aspal, tanpa mengubah secara signifikan ukuran dan bentuk objek utama (lubang jalan) yang berukuran lebih besar dari elemen struktur.

$$B_{close} = B \cdot K_{close} = (B \oplus K_{close}) \ominus K_{close}$$

Keterangan:

B_{close} : Citra biner hasil operasi closing (mask prediksi akhir)

K_{close} : Elemen struktur untuk operasi closing, berbentuk elips (*ellipse*) berukuran 15×15

Operasi closing bertujuan untuk menutup celah atau lubang-lubang kecil yang terdapat di dalam area objek (misalnya akibat pantulan cahaya di tengah lubang jalan yang membuat area tersebut tidak tersegmentasi dengan sempurna), sehingga bentuk objek lubang jalan menjadi lebih utuh dan menyatu.

2.3.6 Ekstraksi Kontur

Ekstraksi kontur (contour extraction) merupakan teknik dalam pengolahan citra digital yang digunakan untuk mengidentifikasi dan menelusuri batas-batas

(boundary) suatu objek pada citra biner, dengan cara menghubungkan titik-titik piksel yang memiliki intensitas sama (umumnya piksel putih/objek pada citra hasil segmentasi) yang berada pada tepi luar objek tersebut. Hasil dari proses ini berupa kumpulan koordinat (x, y) yang membentuk garis batas tertutup mengelilingi setiap objek terpisah pada citra, sehingga objek-objek tersebut dapat dianalisis secara individual, baik dari segi bentuk, posisi, maupun ukurannya.

$$P_{tepi(x,y)} = 1 \Leftrightarrow B_{final(x,y)} = 255\ dan\ \exists(x', y') \in N_{8(x,y)dengan} B_{final(x',y')=0}$$

Keterangan:

$P_{tepi}(x, y)$: Penanda apakah piksel (x, y)merupakan piksel tepi objek

$N_8(x,y)$: Himpunan 8 piksel tetangga (*8-connectivity*) di sekitar koordinat (x, y)

$\exists$: Terdapat (setidaknya satu)

Piksel-piksel tepi yang saling terhubung kemudian ditelusuri secara berurutan mengikuti arah jarum jam (atau berlawanan, tergantung konvensi algoritma) hingga membentuk satu kurva tertutup, yang direpresentasikan sebagai satu kontur:

$$C\_k = \{(x\_0, y\_0), (x\_1, y\_1), \ldots, (x\_(m-1), y\_(m-1))\}$$

dengan syarat batas (*closure condition*):

$$(x_m, y_m) = (x_0, y_0)$$

Keterangan:

C_k : Kontur ke-k hasil penelusuran

(xi,yi) : Titik koordinat ke-i penyusun Kontur

M : Jumlah total titik pada kontur C_k

2.4 OpenCV

OpenCV (Open Source Computer Vision Library) merupakan sebuah pustaka perangkat lunak yang bersifat open source yang digunakan untuk pengolahan citra digital dan pengembangan aplikasi computer vision. OpenCV pertama kali dikembangkan oleh Intel pada tahun 1999 dan hingga saat ini terus dikembangkan oleh komunitas open source di seluruh dunia. Library ini menyediakan berbagai fungsi dan algoritma yang dapat digunakan untuk melakukan berbagai proses pengolahan citra seperti pembacaan gambar, manipulasi citra, segmentasi objek, deteksi tepi, pengenalan pola, hingga analisis video secara real-time (Zulkhaidi, Maria and Yulianto, 2020).

OpenCV dirancang untuk mendukung berbagai bahasa pemrograman seperti C++, Python, dan Java, sehingga banyak digunakan dalam penelitian maupun pengembangan sistem berbasis pengolahan citra. Salah satu keunggulan OpenCV adalah kemampuannya dalam melakukan pemrosesan citra secara efisien dan cepat karena telah dioptimalkan untuk berbagai platform komputasi. Selain itu, OpenCV juga menyediakan banyak fungsi siap pakai yang memudahkan pengembang dalam membangun sistem computer vision tanpa harus membuat algoritma dari awal (Zebua and Rosyani, 2024).

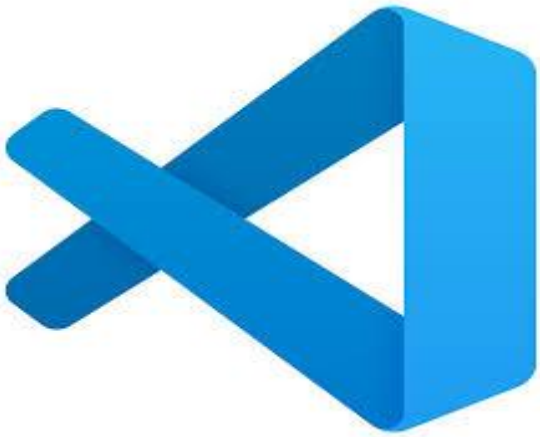
Dalam pengolahan citra digital, OpenCV menyediakan berbagai fungsi penting seperti pembacaan citra, konversi warna, filtering, thresholding, segmentasi citra, serta deteksi kontur. Fungsi-fungsi tersebut memungkinkan sistem untuk mengekstraksi informasi penting dari sebuah citra sehingga dapat digunakan untuk

analisis lebih lanjut. Dengan memanfaatkan berbagai metode yang tersedia pada OpenCV, proses pengolahan citra dapat dilakukan secara lebih efektif dan terstruktur (Supiyandi Supiyandi *et al.*, 2024).

Pada penelitian ini, OpenCV digunakan sebagai library utama dalam proses pengolahan citra untuk mendeteksi lubang jalan. OpenCV digunakan untuk membaca citra input, melakukan proses pre-processing seperti konversi citra dan thresholding, serta melakukan segmentasi objek untuk memisahkan area lubang jalan dari permukaan jalan. Selanjutnya, OpenCV juga digunakan untuk melakukan deteksi kontur pada objek yang teridentifikasi sebagai lubang jalan sehingga luas area lubang dapat dihitung berdasarkan jumlah piksel yang terdeteksi. Dengan memanfaatkan OpenCV, sistem dapat melakukan proses analisis citra secara otomatis sehingga mempermudah proses identifikasi dan pengukuran luas lubang jalan.

2.5 Visual Studio Code

Visual Studio Code merupakan sebuah perangkat lunak source code editor yang dikembangkan oleh Microsoft. Visual Studio Code pertama kali dirilis pada tahun 2015 dan dirancang sebagai editor kode yang ringan namun memiliki fitur yang lengkap untuk mendukung berbagai bahasa pemrograman. Perangkat lunak ini bersifat open source dan dapat digunakan pada berbagai sistem operasi seperti Windows, Linux, dan macOS (Murani *et al.*, 2025).



Gambar 2.2 Logo visual Studio Code

Visual Studio Code menyediakan berbagai fitur yang mendukung proses pengembangan perangkat lunak, seperti penyorotan sintaks (syntax highlighting), pelengkapan kode otomatis (auto-completion), debugging, integrasi dengan sistem kontrol versi seperti Git, serta dukungan berbagai ekstensi yang dapat memperluas fungsi editor. Dengan adanya fitur-fitur tersebut, pengembang dapat menulis, mengedit, serta mengelola kode program dengan lebih mudah dan efisien. Selain itu, Visual Studio Code juga mendukung berbagai bahasa pemrograman populer seperti Python, JavaScript, C++, dan Java. Dukungan tersebut diperoleh melalui berbagai ekstensi yang dapat diinstal sesuai kebutuhan pengguna. Dalam pengembangan aplikasi berbasis Python, Visual Studio Code menyediakan ekstensi khusus yang memungkinkan pengguna untuk menjalankan program, melakukan debugging, serta mengelola pustaka yang digunakan dalam proyek (Abidah, Hadi Wijoyo and Rahman, 2025).

Pada penelitian ini, Visual Studio Code digunakan sebagai lingkungan pengembangan (development environment) untuk menulis dan menjalankan kode program yang digunakan dalam sistem deteksi dan pengukuran luas lubang jalan.

Melalui Visual Studio Code, proses penulisan kode Python, pengolahan citra menggunakan OpenCV, serta pengujian program dapat dilakukan secara lebih terstruktur dan efisien. Dengan dukungan fitur yang lengkap, Visual Studio Code membantu mempercepat proses pengembangan dan mempermudah proses pengujian sistem yang dibangun.

2.6 Phyton

Phyton merupakan salah satu bahasa pemrograman tingkat tinggi (high-level programming language) yang banyak digunakan dalam pengembangan perangkat lunak, analisis data, kecerdasan buatan, serta pengolahan citra digital. Bahasa pemrograman ini pertama kali dikembangkan oleh Guido van Rossum pada tahun 1991 dan dirancang dengan tujuan agar mudah dipelajari serta memiliki sintaks yang sederhana dan mudah dibaca. Python bersifat open source, sehingga dapat digunakan secara bebas oleh siapa saja dan terus dikembangkan oleh komunitas pengembang di seluruh dunia (Buwono *et al.*, 2025).

Python memiliki berbagai keunggulan dibandingkan dengan bahasa pemrograman lainnya, di antaranya adalah sintaks yang sederhana, dukungan pustaka yang sangat banyak, serta kompatibilitas dengan berbagai sistem operasi seperti Windows, Linux, dan macOS. Selain itu, Python juga mendukung berbagai paradigma pemrograman seperti pemrograman berorientasi objek (object-oriented programming), pemrograman prosedural, dan pemrograman fungsional. Hal ini menjadikan Python sebagai salah satu bahasa pemrograman yang sangat fleksibel untuk berbagai kebutuhan pengembangan sistem. Dalam bidang pengolahan citra digital dan computer vision, Python banyak digunakan karena memiliki berbagai

pustaka yang mendukung proses analisis citra, seperti OpenCV, NumPy, dan Matplotlib. Pustaka-pustaka tersebut menyediakan berbagai fungsi yang dapat digunakan untuk membaca citra, memproses data piksel, melakukan segmentasi objek, serta melakukan visualisasi hasil pengolahan citra (Adawiyah Ritonga and Yahfizham Yahfizham, 2023).

Pada penelitian ini, Python digunakan sebagai bahasa pemrograman utama dalam pengembangan sistem deteksi dan pengukuran luas lubang jalan. Python digunakan untuk mengintegrasikan berbagai pustaka pengolahan citra, mengolah data citra input, serta menampilkan hasil analisis melalui antarmuka aplikasi. Dengan menggunakan Python, proses pengembangan sistem menjadi lebih efisien karena banyak fungsi pengolahan citra yang telah tersedia dalam bentuk pustaka sehingga mempermudah proses implementasi metode yang digunakan.

2.7 Pengujian Confusion Matrik

Pengujian Confusion Matrix merupakan salah satu metode evaluasi yang digunakan untuk mengukur kinerja suatu sistem klasifikasi atau deteksi dengan membandingkan hasil prediksi model dengan data sebenarnya (ground truth). Metode ini banyak digunakan dalam bidang pengolahan citra, machine learning, maupun sistem deteksi objek untuk mengetahui seberapa baik sistem dalam mengidentifikasi objek yang benar. Confusion matrix biasanya disajikan dalam bentuk tabel yang terdiri dari empat komponen utama, yaitu True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN). True Positive merupakan kondisi ketika sistem berhasil mendeteksi objek dengan benar sesuai dengan data sebenarnya. False Positive terjadi ketika sistem mendeteksi objek yang

sebenarnya tidak ada atau bukan objek yang dimaksud. False Negative merupakan kondisi ketika objek sebenarnya ada, namun sistem gagal mendeteksinya. Sedangkan True Negative adalah kondisi ketika sistem dengan benar mengidentifikasi bahwa tidak terdapat objek pada area tersebut (Yovi Apridiansyah *et al.*, 2024).

Berdasarkan nilai-nilai yang terdapat pada confusion matrix, dapat dihitung beberapa metrik evaluasi untuk mengetahui performa sistem secara lebih detail. Beberapa metrik yang umum digunakan antara lain Accuracy, Precision, Recall, dan F1-Score. Accuracy digunakan untuk mengukur tingkat ketepatan keseluruhan sistem dalam melakukan klasifikasi. Precision digunakan untuk mengetahui seberapa banyak prediksi positif yang benar-benar merupakan objek yang benar. Recall digunakan untuk mengukur kemampuan sistem dalam menemukan seluruh objek yang sebenarnya ada. Sedangkan F1-Score merupakan nilai rata-rata harmonis dari precision dan recall yang digunakan untuk memberikan gambaran keseimbangan antara kedua metrik tersebut.

Dalam penelitian ini, pengujian confusion matrix dilakukan dengan membandingkan mask hasil deteksi sistem dengan mask ground truth pada setiap citra dataset. Perbandingan tersebut dilakukan pada tingkat piksel sehingga dapat diketahui jumlah piksel yang termasuk dalam kategori TP, FP, FN, dan TN. Hasil perhitungan tersebut kemudian digunakan untuk menghitung nilai evaluasi yang menggambarkan tingkat keberhasilan sistem dalam mendeteksi lubang jalan serta menghitung luas area lubang secara lebih akurat. Dengan menggunakan metode confusion matrix, performa sistem deteksi lubang jalan dapat dianalisis secara lebih

objektif dan terukur. Evaluasi ini sangat penting untuk mengetahui seberapa baik metode pengolahan citra yang digunakan dalam penelitian ini dalam mengidentifikasi dan mengukur luas lubang jalan pada citra permukaan jalan.