

# Laporan Praktikum Kontrol Cerdas

Nama : Fryscha Febryanti Puspitasari  
NIM : 224308008  
Kelas : TKA 6A  
Akun Github (Tautan) : <https://github.com/Fryscha>  
Student Lab Assistant : Yulia Brilianty

## 1. Judul Percobaan

Deteksi Kerusakan dan Gangguan pada Jalur Rel Kereta Api PNM menggunakan Yolov8

## 2. Tujuan Percobaan

Percobaan ini dilakukan dengan tujuan sebagai berikut.

- Memahami konsep deteksi kerusakan dan gangguan pada jalur rel kereta api.
- Mengimplementasikan YOLOv8 sebagai model deteksi objek real-time
- Membuat dataset *object detection* untuk eksperimen.
- Mengidentifikasi dan mengklasifikasi jenis kerusakan atau gangguan pada rel kereta api

## 3. Landasan Teori

Perawatan jalan rel meliputi pemeriksaan kondisi jalan rel dan penyusunan program perawatan. Pemeriksaan kondisi rel dilakukan sebagai tindakan awal untuk memperoleh data mengenai kondisi suatu petak lintas. Kondisi geometri rel yang baik sangat diperlukan untuk keamanan dan kenyamanan perjalanan kereta api. Beban lintas atau akibat peristiwa alam akan mengakibatkan perubahan bentuk geometri rel kereta api (Muhtarom & Ratih, 2021)

Deteksi kondisi rel kereta api, seperti kerusakan dan gangguan, merupakan salah satu kegiatan penting dalam perawatan rel. Hal ini dilakukan untuk menjaga keselamatan, serta kelancaran operasional transportasi perkeretaapian. Dengan kemajuan teknologi pengolahan citra digital dan kecerdasan buatan, pendekatan otomatis menggunakan algoritma deteksi

objek menjadi solusi yang efisien. Metode CNN memiliki banyak cabang yang dikembangkan seiring dengan berkembangnya teknologi *computer vision* dan algoritma *deep learning*, salah satu metode yang populer saat ini adalah *You Only Look Once* (YOLO) (Erwi & Irsyad, 2024).

YOLO memiliki kapabilitas yang terbaik untuk mengenali objek dengan akurasi yang tinggi dan kecepatan waktu deteksi, salah satunya adalah YOLOv8 (Bhavana et al., 2024). YOLOv8 merupakan pengembangan terbaru dari seri model deteksi objek YOLO yang memanfaatkan kemajuan teknologi masa kini untuk memungkinkan mesin belajar mengenali objek layaknya manusia (Saepudin *et al.*, 2024). Sebagai model *single-stage detector*, YOLOv8 memungkinkan deteksi *real-time* terhadap berbagai jenis kerusakan seperti retakan dan gangguan di jalur rel. Dengan model pada dataset citra rel kereta yang relevan, YOLOv8 dapat mengidentifikasi dan mengklasifikasikan gangguan secara otomatis, yang pada akhirnya membantu meningkatkan efektivitas sistem monitoring dan pemeliharaan jalur kereta api (Zhou et al., 2024).

RoboFlow merupakan platform web yang berfungsi untuk pengelolaan dataset secara komprehensif, mulai dari pengumpulan hingga optimasi data untuk pengembangan model *computer vision*. Platform ini menyediakan berbagai alat yang mempermudah proses *preprocessing*, seperti augmentasi data, pengubahan ukuran gambar, anotasi, dan pengelolaan dataset secara kolaboratif. Selain itu, RoboFlow juga mendukung pengembangan model dengan teknik augmentasi otomatis dan konversi dataset ke format yang kompatibel dengan berbagai *framework Machine Learning*, termasuk TensorFlow, PyTorch, dan YOLO. Dengan fitur-fitur tersebut, RoboFlow menjadi kerangka kerja yang membantu para pengembang meningkatkan efisiensi, kualitas, dan akurasi model *computer vision* secara keseluruhan (Hayati *et al.*, 2023).

Dalam pengembangan sistem *object detection*, kualitas dataset yang digunakan sangat menentukan kinerja model yang dihasilkan. Dataset tersebut mencakup gambar-gambar objek dalam bentuk *bounding box*. *Bounding box* adalah kotak persegi panjang yang mengelilingi objek dalam gambar, memberikan informasi spasial yang memungkinkan model untuk belajar mengenali dan membedakan objek secara akurat. Anotasi yang presisi dan konsisten sangat penting karena kesalahan kecil dalam penandaan dapat berdampak signifikan terhadap performa model, terutama dalam metrik evaluasi seperti *Intersection over Union* (IoU) dan *Average Precision* (AP). Oleh karena itu, proses anotasi harus dilakukan dengan cermat, menggunakan alat bantu yang sesuai dan melibatkan pelabel yang terlatih,

untuk memastikan bahwa model dapat belajar secara efektif dan memberikan hasil prediksi yang andal dalam berbagai kondisi nyata (Murrugarra-Llerena et al., 2022).

## 4. Analisis dan Diskusi

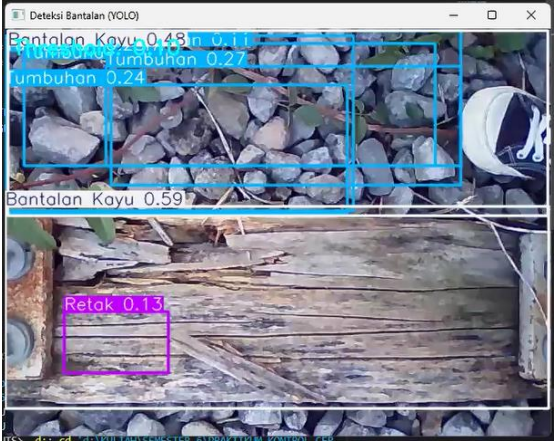
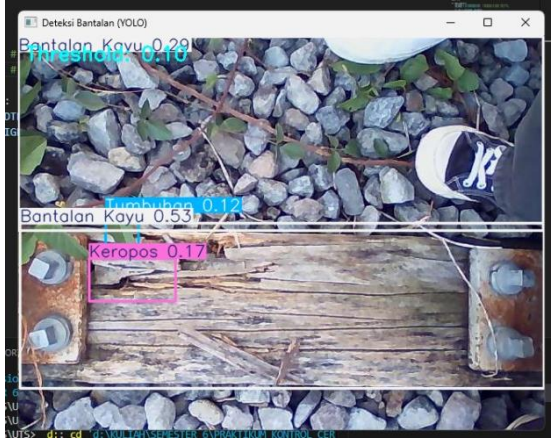

Praktikum ini menunjukkan bahwa penggunaan YOLOv8 sebagai model deteksi objek mampu memberikan solusi efektif untuk mendeteksi kerusakan dan gangguan pada jalur rel kereta api secara otomatis. Dengan kemampuan *real-time detection*, YOLOv8 dapat mengidentifikasi berbagai jenis objek seperti retakan, pelapukan, dan benda asing dengan akurasi yang cukup baik.

Namun demikian, akurasi model sangat bergantung pada kualitas dan jumlah data pelatihan. Dataset yang terbatas dan kurang bervariasi dapat menyebabkan model kesulitan mengenali objek dalam kondisi lingkungan yang berbeda. Oleh karena itu, proses pengumpulan data, pelabelan yang akurat, serta augmentasi data menjadi faktor penting dalam keberhasilan model.

## 5. Assignment

Membuat deteksi kerusakan dan gangguan pada rel kereta api dengan sistem 2 kamera menggunakan YOLOv8. Diasumsikan 1 kamera menghadap bantalan untuk mendeteksi kerusakan dan gangguan pada bantalan dan kamera yang lain menghadap ke arah depan untuk mendeteksi gangguan pada rel kereta.

## 6. Data dan Output Hasil Pengamatan

No	Variabel	Hasil Pengamatan
1	Bantalan Retak	
2	Bantalan Keropos	
3	Bantalan Kayu	

4	Bantalan Beton	
5	Person	
6	Tumbuhan	
7	Car	



8	Motorcycle	
---	------------	---

## 7. Kesimpulan

Berdasarkan percobaan dan analisis data dapat disimpulkan bahwa:

- a. Hasil deteksi menunjukkan bahwa YOLOv8 mampu mengenali objek-objek penting seperti halnya retakan bantalan, penghalang jalan, tumbuhan merambat, ataupun material asing yang berada di bantalan, dengan catatan objek tersebut telah dilabeli dengan benar dalam dataset pelatihan.
- b. Performa model cukup baik, tetapi kualitas dataset sangat penting. Hal ini memengaruhi hasil dari deteksi kerusakan dan gangguan. Guna memperkuat performa sistem deteksi secara keseluruhan, peningkatan kuantitas dan variasi data sangat diperlukan, juga pelabelan yang akurat.

## 8. Saran

Disarankan untuk memperluas dan memvariasikan dataset agar model YOLOv8 memiliki akurasi dan kemampuan generalisasi yang lebih baik. Pelabelan data harus dilakukan secara teliti dan konsisten disertai pengecekan kembali untuk memastikan akurasi dan mencegah kesalahan klasifikasi.

## 9. Daftar Pustaka

Bhavana, N., Kodabagi, M. M., Kumar, B. M., Ajay, P., Muthukumaran, N., & Ahilan, A. (2024). POT-YOLO: Real-Time Road Potholes Detection Using Edge Segmentation-Based Yolo V8 Network. *IEEE Sensors Journal*, 24(15), 24802–24809. <https://doi.org/10.1109/JSEN.2024.3399008>

Erwi, S. G. V. K., & Irsyad, H. (2024). Implementasi Deteksi Objek Pada Jalan Rusak menggunakan Metode YOLOv8. *Buletin Ilmiah Informatika Teknologi*, 3.

- Hayati, N. J., Singasatia, D., & Muttaqin, M. R. (2023). Object Tracking Menggunakan Algoritma You Only Look Once (YOLO)v8 untuk Menghitung Kendaraan. *Komputa : Jurnal Ilmiah Komputer dan Informatika*, 12(2), 91–99. <https://doi.org/10.34010/komputa.v12i2.10654>
- Muhtarom, Z., & Ratih, S. Y. (2021). Analisis Kondisi Jalan Rel Kereta Api Pada Lintas Sragen-Solo Berdasarkan Nilai Track Quality Indeks (TQI). *Jurnal Teknik Sipil*, 17(1), 01–13. <https://doi.org/10.28932/jts.v17i1.3440>
- Murrugarra-Llerena, J., Kirsten, L., & Jung, C. R. (2022). Can We Trust Bounding Box Annotations for Object Detection? *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 4812–4821. <https://doi.org/10.1109/CVPRW56347.2022.00528>
- Saepudin, S., Sujana, N., Mutoffar, M. M., & Haryanto, A. A. (2024). Analisis Kinerja Yolov8 Optimalisasi Roboflow Untuk Deteksi Ekspresi Wajah Emosional Dengan Machine Learning. *Naratif: Jurnal Nasional Riset, Aplikasi dan Teknik Informatika*, 6(2), 115–124. <https://doi.org/10.53580/naratif.v6i2.292>
- Zhou, Q., Wang, Z., Zhong, Y., Zhong, F., & Wang, L. (2024). Efficient Optimized YOLOv8 Model with Extended Vision. *Sensors*, 24(20), 6506. <https://doi.org/10.3390/s24206506>

# Lampiran

## 1. Program Yolo Detector

```
from ultralytics import YOLO

import cv2

# Load model sekali saja
model = YOLO("best.pt") # Ganti dengan model custom kalau ada

def detect_objects(frame, conf=0.3):
    results = model.predict(source=frame, conf=conf, verbose=False)
    annotated_frame = results[0].plot()
    return annotated_frame, results[0].boxes.data # Bisa dipakai untuk logika lebih lanjut
```

## 2. Program Camera

```
import cv2

def open_cameras():
    cam_bawah = cv2.VideoCapture(2) # Menghadap ke bawah (bantalan)
    cam_depan = cv2.VideoCapture(0) # Menghadap ke depan (jalur rel)

    for cam in [cam_bawah, cam_depan]:
        cam.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

    return cam_bawah, cam_depan
```

## 3. Program Deteksi

```
import cv2

from yolo_detector import detect_objects
from camera_utils import open_cameras
```



```
# Nilai awal threshold (dalam % karena trackbar butuh int)
```

```
initial_threshold = 50 # 50% = 0.5
```

```
def nothing(x):
```

```
    pass # Fungsi dummy untuk trackbar
```

```
def proses_deteksi(kamera, nama_window, conf_threshold):
```

```
    ret, frame = kamera.read()
```

```
    if not ret:
```

```
        print(f"Gagal membaca frame dari {nama_window}")
```

```
        return False
```

```
    hasil_frame, _ = detect_objects(frame, conf=conf_threshold)
```

```
    # Tampilkan nilai threshold
```

```
    cv2.putText(
```

```
        hasil_frame,
```

```
        f"Threshold: {conf_threshold:.2f}",
```

```
        (10, 30),
```

```
        cv2.FONT_HERSHEY_SIMPLEX,
```

```
        0.8,
```

```
        (255, 255, 0),
```

```
        2
```

```
    )
```

```
    cv2.imshow(nama_window, hasil_frame)
```

```
    return True
```

```
def main():
```

```
    cam_bawah, cam_depan = open_cameras()
```

```
    if not (cam_bawah.isOpened() and cam_depan.isOpened()):
```

```
print("Gagal membuka salah satu kamera.")
```

```
return
```

```
# Buat jendela dan slider trackbar
```

```
cv2.namedWindow("Kontrol Threshold")
```

```
cv2.createTrackbar("Confidence %", "Kontrol Threshold", initial_threshold, 100, nothing)
```

```
while True:
```

```
    # Ambil nilai threshold dari slider (0 - 100), lalu ubah ke float (0.0 - 1.0)
```

```
    conf = cv2.getTrackbarPos("Confidence %", "Kontrol Threshold") / 100.0
```

```
    if not proses_deteksi(cam_bawah, "Deteksi Bantalan (YOLO)", conf):
```

```
        break
```

```
    if not proses_deteksi(cam_depan, "Deteksi Jalur Rel (YOLO)", conf):
```

```
        break
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        break
```

```
cam_bawah.release()
```

```
cam_depan.release()
```

```
cv2.destroyAllWindows()
```

```
if __name__ == "__main__":
```

```
    main()
```