Deteksi Kerusakan dan Gangguan pada Bantalan Rel Kereta Api PNM menggunakan YOLOv8n

Alam Syah Putra Aulia[*1], Fryscha Febryanti Puspitasari[*2], Permata Luthfiya Akram[*3], Tegar Lalang Jagad[*4], Wafa Nabila Ulima Khoirunnisa[*5], Widhi Widya Hastuti[*6]

[#]*Teknik Perkeretaapian, Politeknik Negeri Madiun*
Jl. Ring Road Barat, Winongo, Kec. Manguharjo, Kota Madiun, Jawa Timur

[1]alamsyahputraaulia@gmail.com
[2]fryschafebryanti25@gmail.com
[3]permataakram.pa@gmail.com
[4]tegarlj12@gmail.com
[5]khoirunnisawafa28@gmail.com
[6]widhiwidya08@gmail.com

***Abstract*– The object and disturbance detection process on PNM railroad sleepers employs the sleepers as an object and disturbance detection model that uses the YOLOv8n algorithm. Images as well as videos featuring sleepers together with objects start the process. Then, Roboflow is used to annotate the data for each class. Classes include things like wooden bearings together with concrete bearings as well as cracks that are porous, also people, plus plants, with cars, in addition to motorcycles. Model training in Google Colab then followed this so that it could produce a detection model ready for use. The training results show about the ability of the model to recognize various objects with their respective accuracy levels. These levels of accuracy include cracks at 13%, porousness at 17%, wooden sleepers at 47%, concrete sleepers at 59%, people at 13%, plants at 14%, cars at 36%, and even motorcycles at 74%. YOLOv8n can detect different types of damage to sleepers plus disturbing railway objects, if objects are labeled well in the training dataset.**

***Keywords*– *Damage, Detection, Disturbance, Railways, Sleepers,* YOLOv8n**

***Abstrak* – Proses deteksi objek dan gangguan pada bantalan rel kereta api PNM menggunakan algoritma YOLOv8n sebagai model deteksi objek serta gangguan pada bantalan. Proses tersebut dimulai dengan pengumpulan dataset berupa gambar serta video dari bantalan beserta beberapa objek di sekitarnya. Selanjutnya, data tersebut dianotasi sesuai kelas masing-masing menggunakan Roboflow, seperti bantalan kayu, bantalan beton, retakan, keropos, orang, tumbuhan, mobil, serta motor. Dilanjutkan dengan pelatihan model di Google Colab untuk menghasilkan suatu model deteksi yang siap pakai. Hasil pelatihan memperlihatkan kemampuan model untuk mengenali beragam objek dengan tingkat akurasi masing-masing, yaitu retak sebesar 13%, keropos sebesar 17%, bantalan kayu sebesar 47%, bantalan beton sebesar 59%, orang sebesar 13%, tumbuhan sebesar 14%, mobil sebesar 36%, dan motor sebesar 74%, seperti yang ditunjukkan oleh hasil pelatihan. YOLOv8n mampu untuk mendeteksi berbagai jenis kerusakan pada bantalan serta objek lain yang mengganggu di rel kereta, berdasarkan hasil itu, asalkan objek-objek itu diberi label secara tepat dalam dataset pelatihan.**

***Kata kunci*– Bantalan, Deteksi, Gangguan, Kerusakan, Rel, YOLOv8n**

I. PENDAHULUAN

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 [1].

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) [2].

YOLO memiliki kapabilitas yang terbaik untuk mengenali objek dengan akurasi yang tinggi dan kecepatan waktu deteksi, salah satunya adalah YOLOv8 [3]. YOLOv8 merupakan pengembangan terbaru dari seri model deteksi objek YOLO yang memanfaatkan kemajuan teknologi masa kini untuk memungkinkan mesin belajar mengenali objek layaknya manusia [4]. 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 [5].

RoboFlow merupakan platform 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 [6].

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.

II. METODE

Praktikum deteksi kerusakan bantalan menggunakan algoritma YOLOv8n untuk mendeteksi objek. YOLO (*You Only Look Once*) adalah salah satu metode deteksi objek secara *real-time*. Secara umum, YOLO membagi gambar menjadi beberapa bagian, lalu setiap bagian diprediksi apakah terdapat objek didalamnya, di mana letak objek, dan jenis objeknya.

A. Pengumpulan Dataset

Langkah pertama dalam proses ini adalah mengumpulkan dataset berupa gambar atau video yang merekam kondisi permukaan rel kereta api. Data ini dapat diperoleh dari berbagai sumber, seperti hasil perekaman langsung di lapangan atau *database* publik. Gambar yang dikumpulkan berupa kondisi bantalan kereta api dan jalur rel kereta api.

B. Anotasi Data

Tahapan berikutnya yaitu anotasi data menggunakan Roboflow. Tujuan dari anotasi data ini adalah untuk memberikan tanda berupa *bounding box* yang menunjukkan koordinat posisi objek tersebut. Selain itu, setiap *bounding box* juga terdapat informasi kelas objek atau label yang digunakan untuk mengidentifikasi jenis objek tersebut serta nilai akurasi untuk keakuratan objek yang dideteksi.

C. Pelatihan Model

Setelah data dianotasi, langkah selanjutnya yaitu melatih dataset menggunakan Roboflow dengan pengaturan dataset 69% *Train Set*, 15% *Valid Set*, 16% *Test Set.*

Tahap pelatihan model dilakukan di platform Google Colab dan mencakup beberapa langkah penting, mulai dari instalasi pustaka, impor dataset, hingga pelatihan dataset. Langkah awal berupa pengecekan perangkat GPU/TPU pada Google Colab guna memastikan pelatihan berjalan secara optimal. Selanjutnya instalasi pustaka *ultralytics* yang digunakan untuk menjalankan YOLOv8n serta pustaka Roboflow untuk mengimpor dataset langsung dari *platform* Roboflow.

Selanjutnya dilakukan dengan menetapkan jumlah epochs sebanyak 50 (epochs=50) dan ukuran gambar *(image size)* sebesar 509 piksel.

III. HASIL DAN PEMBAHASAN

Dari pelatihan model menggunakan Google Colab, didapatkan data berupa beberapa grafik. Selanjutnya, dari pengujian secara *real-time* menunjukkan bahwa program mampu mendeteksi beberapa objek dan gangguan pada bantalan rel.

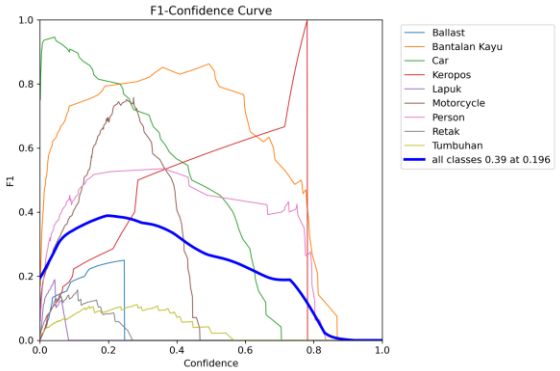
A. Hasil Pelatihan Model

Berikut merupakan data dari beberapa grafik yang didapat dari pelatihan model menggunakan Google Colab, terlihat pada **Tabel 3.1**.

Tabel 3.1 Pelatihan Model

Metrik	Nilai dari Grafik
Box Loss (train/vall)	1.4/2.3
Classification Loss	1.0/2.1
Distribution Loss	0.3/0,5
Precision	55%
Recall	43%
mAP@0.5	43%
mAP@0.5:0.95	21%

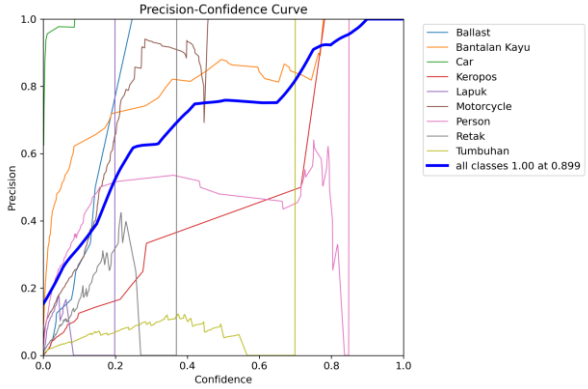
Hasil dari tabel ini diambil berdasarkan pembacaan grafik di bawah ini.



Gambar 3.1 Grafik F1-*Confidence Curve*

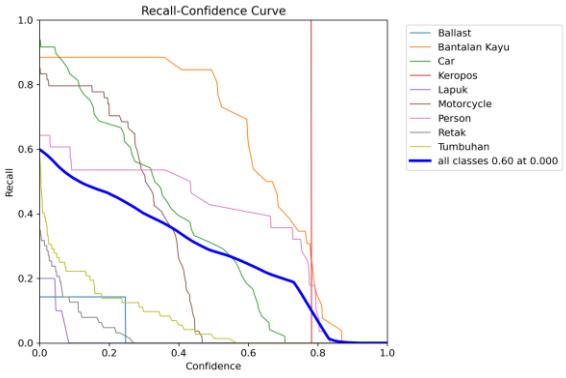
F1-*Confidence curve* menunjukkan hubungan antara precision dengan skor F1 (*recall*) berdasarkan *threshold confidence*. Dari grafik terlihat bahwa titik F1 terbaik untuk semua kelas terjadi di *confidence* 0.196 dengan skor F1 sebesar 0.39.

Keropos, Bantalan Kayu, dan *Car* menunjukkan skor F1 yang tinggi, berarti model memiliki keseimbangan *precision* dan *recall* yang baik. Namun sebaliknya skor F1 rendah untuk kelas seperti Lapuk, Retak, dan Tumbuhan, menandakan ketidakseimbangan atau kurangnya data.



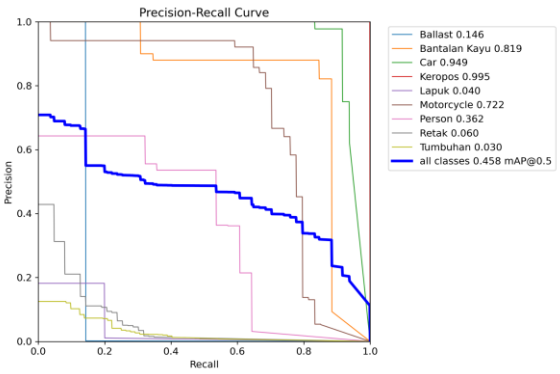
Gambar 3.2 Grafik *Precision-Confidence Curve*

Precison-Confidence Curve menunjukkan hubungan antara *confidence* dan *precision*. *Precision* akan cenderung meningkat dengan *confidence*. Model dapat mencapai *precision* 1.00 saat *confidence* bernilai 0.899, namun *recall* kemungkinan akan menurun. Kelas dengan *precision* tinggi pada *confidence* tinggi yaitu *Car* dan Keropos, sedangkan Lapuk, Retak, dan Tumbuhan sangat rendah di hampir semua nilai *confidence*.



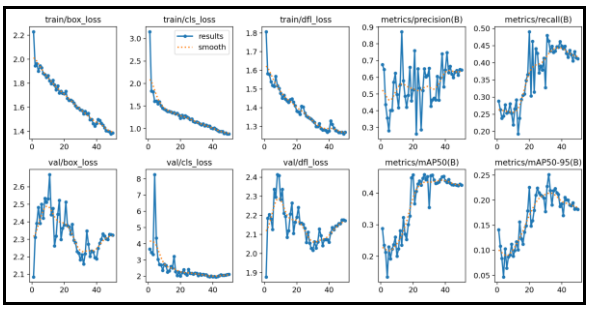
Gambar 3.3 Grafik *Recall-Confidence Curve*

Recall-Confidence Curve menggambarkan seberapa baik model mengenali objek saat *confidence threshold* bervariasi. *Recall* memiliki nilai tertinggi 0.60 pada saat *confidence* bernilai 0.00. Model cenderung lebih baik mengenali Bantalan Kayu, *Car*, dan *Motorcycle* dibandingkan dengan kelas lain.



Gambar 3.4 Grafik *Precision-Recall Curve*

Precision-Recall Curve menunjukkan *trade-off* antara *precision* dan *recall* untuk setiap kelas. Kelas dengan kinerja tinggi yaitu Keropos (0.995), *Car* (0.949), dan Bantalan Kayu (0.819), sedangkan kelas dengan kinerja sedang yaitu *Motorcycle* (0.722) dan *Person* (0.362), ada juga kelasa dengan kinerja rendah yaitu *Ballast* (0.146), Retak (0.060), Lapuk (0.040), dan Tumbuhan (0.030). Nilai mAP@0.5 (mean *Average Precision* pada *threshold* IoU 0.5) sebesar 0.458 untuk semua kelas.



Gambar 3.5 Grafik *Training Loss* & mAP

Grafik Menunjukkan performa model deteksi objek selama proses pelatihan dan validasi dalam 50 epoch. Pada bagian *train loss*, grafik train/box_loss, train/cls_loss, dan train/dfl_loss menunjukkan penurunan yang konsisten dari angka awal diatas 2.0 menuju kisaran 1.3-1.4 menandakan model semakin baik dalam mengenali objek. Namun pada bagian val/box_loss, val/cls_loss, dan val/dfl_loss terlihat fluktuasi yang lebih tinggi dibandingkan *train loss*. val/cls_loss sempat melonjak hingga 8 pada awal pelatihan, model mengalami penurunan yang drastis akan tetapi seiring waktu mengalami perbaikan.

Dari sisi metrik evaluasi, metrics/precision(B) memiliki nilai awal antara 0.7-0.6 dan mencapai nilai maksimum sekitar 0.55. Hal ini bisa disebabkan oleh prediksi positif yang cukup banyak tapi tidak selalu benar. Metrics/recall(B) meningkat stabil dari 0.30 mencapai nilai sekitar 0.40–0.45, menandakan bahwa model semakin banyak menangkap objek yang benar. Metrics/mAP50(B) mengalami peningkatan nilai

maksimum 0,43 yang nilai awalnya kurang dari 0.3, dan mAP50-95(B) mencapai nilai 0.21 dengan nilai awal kurang dari 0.15, yang menunjukkan bahwa performa deteksi model masih terbatas ketika dituntut untuk presisi tinggi pada berbagai *threshold* IoU.

B. Hasil Pengujian

Berikut merupakan beberapa objek dan gangguan pada bantalan rel yang terdeteksi menggunakan program secara *real-time* dengan bantuan kamera tambahan.

Gambar 3.6 Retak

Terlihat dari **Gambar 3.6**, objek bantalan yang memiliki cacat keretakan yang terdeteksi dengan akurasi sebesar 13%. Jenis retakan ini terjadi sepanjang bantalan kayu. Hal ini disebabkan oleh material yang menua, paparan cuaca ekstrem, disertai dengan beban berulang dari lalu lintas kereta. Hal ini menunjukkan bahwa bantalan kayu tersebut berada dalam keadaan degradasi struktural yang serius.

Gambar 3.7 Keropos

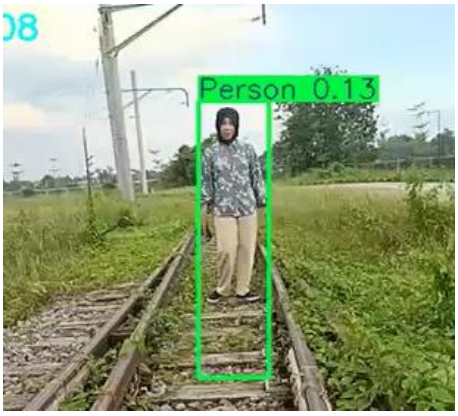
Terlihat dari **Gambar 3.7**, objek bantalan keropos yang terdeteksi dengan akurasi sebesar 17% yang merupakan hasil dari proses pelapukan alami kayu karena kelembapan yang tinggi, mikroorganisme seperti jamur atau serangga rayap, dan paparan cuaca jangka panjang. Struktur kayu yang ada pada gambar 3.7 menunjukkan struktur yang telah kehilangan jumlah kepadatan dan kekuatan yang cukup besar, yang menyebabkan kelemahan kapasitas penyangga dari ballast di bawah rel dan kereta.

Gambar 3.8 Bantalan Kayu

Terlihat dari **Gambar 3.8**, objek bantalan kayu terdeteksi memiliki akurasi sebesar 47%. Bantalan kayu pada rel kereta memiliki bentuk balok persegi panjang, berwarna cokelat keabu-abuan akibat pelapukan, dan terbuat dari material kayu yang kuat. Bantalan ini dilengkapi plat besi dan baut pengikat di kedua sisi rel untuk menjaga kestabilan.

Gambar 3.9 Bantalan Beton

Terlihat dari **Gambar 3.9**, objek bantalan beton terdeteksi memiliki akurasi sebesar 59%. Bantalan beton berfungsi untuk menopang rel kereta, memastikan jarak antar rel dan kestabilan antar rel serta membagi beban kereta ke lapisan *ballast* (kerikil) dengan tepat. Bantalan tersebut dalam keadaan yang baik, hanya saja terdapat flora (tumbuhan) yang mengganggu struktur ballast apabila dibiarkan dalam waktu yang relatif lama.



Gambar 3.10 *Person* (Orang)

Terlihat dari **Gambar 3.10**, objek *person* atau manusia yang terdeteksi dengan akurasi 13% yang berada di atas rel, dimana kondisi ini sangat beresiko dan membahayakan keselamatan terutama pada saat jalur kereta digunakan.

Gambar 3.11 Tumbuhan

Terlihat dari **Gambar 3.11**, objek tumbuhan terdeteksi memiliki akurasi 14%. Pertumbuhan tanaman yang berada di rel kereta api menunjukkan bahwa adanya kekurangan pemeliharaan rel kereta api yang dalam jangka panjang dapat mengancam integritas rel karena akar dapat mendorong atau menggeser batu *ballast*. Oleh karena itu, meskipun tampak kecil dan remeh, pertumbuhan vegetasi harus dikendalikan secara berkala sebagai bahan dari pemeliharaan rutin rel.

Gambar 3.12 *Car* (Mobil)

Terlihat dari **Gambar 3.12**, objek Mobil terdeteksi oleh sistem dengan label "*Car*" dan memiliki akurasi sebesar 36%. Mobil ini juga tampak menghadap kamera dari kejauhan dan berada di area yang kemungkinan merupakan bagian dari fasilitas.

Gambar 3.13 *Motorcycle* (Motor)

Terlihat dari **Gambar 3.13**, objek motor terdeteksi oleh sistem dengan label "*motorcycle*" memiliki akurasi sebesar 74%. Posisi motor yang terletak cukup dekat dengan rel menunjukkan bahwa area ini merupakan fasilitas yang menyediakan tempat parkir bagi pengunjung atau pegawai.

IV.KESIMPULAN

Hasil deteksi menunjukkan bahwa YOLOv8n mampu mengenali objek seperti halnya bantalan keropos, retakan bantalan, penghalang jalan, tumbuhan merambat, ataupun material asing yang berada di bantalan, dengan catatan objek tersebut telah dilabeli dengan benar dalam dataset pelatihan.

REFERENSI

[1] Z. Muhtarom and S. Y. Ratih, "Analisis Kondisi Jalan Rel Kereta Api Pada Lintas Sragen-Solo Berdasarkan Nilai Track Quality Indeks (TQI)," *J. Tek. Sipil*, vol. 17, no. 1, pp. 01–13, Mar. 2021, doi: 10.28932/jts.v17i1.3440.

[2] S. G. V. K. Erwi and H. Irsyad, "Implementasi Deteksi Objek Pada Jalan Rusak menggunakan Metode YOLOv8," *Bul. Ilm. Inform. Teknol.*, vol. 3, 2024.

[3] N. Bhavana, M. M. Kodabagi, B. M. Kumar, P. Ajay, N. Muthukumaran, and A. Ahilan, "POT-YOLO: Real-Time Road Potholes Detection Using Edge Segmentation-Based Yolo V8 Network," *IEEE Sens. J.*, vol. 24, no. 15, pp. 24802–24809, Aug. 2024, doi: 10.1109/JSEN.2024.3399008.

[4] S. Saepudin, N. Sujana, M. M. Mutoffar, and A. A. Haryanto, "Analisis Kinerja Yolov8 Optimalisasi Roboflow Untuk Deteksi Ekspresi Wajah Emosional Dengan Machine Learning," *Naratif J. Nas. Ris. Apl. Dan Tek. Inform.*, vol. 6, no. 2, pp. 115–124, Dec. 2024, doi: 10.53580/naratif.v6i2.292.

[5] Q. Zhou, Z. Wang, Y. Zhong, F. Zhong, and L. Wang, "Efficient Optimized YOLOv8 Model with Extended Vision," *Sensors*, vol. 24, no. 20, p. 6506, Oct. 2024, doi: 10.3390/s24206506.

[6] N. J. Hayati, D. Singasatia, and M. R. Muttaqin, "Object Tracking Menggunakan Algoritma You Only Look Once (YOLO)v8 untuk Menghitung Kendaraan," *Komputa J. Ilm. Komput. Dan Inform.*, vol. 12, no. 2, pp. 91–99, Nov. 2023, doi: 10.34010/komputa.v12i2.10654.

LAMPIRAN

Program YOLO Detector

```
from ultralytics import YOLO
import cv2

model = YOLO("best.pt")

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
```

Program Camera

```
import cv2

def open_cameras():
cam_bawah = cv2.VideoCapture(2)  # Menghadap ke bawah (bantalan)
cam_depan = cv2.VideoCapture(1)  # 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
```

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()
```

Dokumentasi Kegiatan