Real-Time Vision-Based Blind Spot Detection and Tracking on a Raspberry Pi Using Lightweight Deep Learning

M. Samba Siva Reddy*, R. Keerthi Sai Sanjana, K. Susmitha, and Ch. Rohith

Lakireddy Bali Reddy College of Engineering, Mylavaram, 521230, India
sambasivareddymula@gmail.com*,ramisettykeerthi2005@gmail.com,
eshmithareddy9924@gmail.com,rohithchundru872@gmail.com

Abstract. Blind spot-related accidents remain a significant safety risk, particularly with cost-sensitive Advanced Driver Assistance Systems (ADAS), where expensive sensing solutions may not always be practical. In this research paper, a real-time vision-based system is constructed using Raspberry Pi platform for vehicle detection and tracking in blind spots using lightweight YOLOv4-Tiny object recognition to ensure maximum performance on limited hardware resources. For improved reliability, centroid-based tracking with motion data can be used to identify vehicles entering a blind spot region. Alerts are generated only if vehicles remain within this pre-defined zone for an extended period, helping reduce false warnings. This system uses a Raspberry Pi, camera module, LCD display and buzzer to give drivers visual and auditory feedback. Experimental results obtained in urban traffic scenarios demonstrate that this system can achieve consistent detection and tracking despite computational constraints, without needing additional hardware acceleration. Overall, this approach demonstrates how an effective and affordable blind spot monitoring system can be created using lightweight deep learning techniques on embedded platforms.

Keywords: Blind spot detection, advanced driver assistance systems, deep learning, object tracking, Raspberry Pi, embedded vision

1 Introduction

Advanced Driver Assistance Systems (ADAS) play an essential part in increasing road safety by helping drivers make better decisions and preventing accidents from occurring when nearby vehicles aren't visible through mirrors, especially during lane changes or heavy traffic[7, 8]. Blind spot detection plays an especially vital role here by helping prevent blind spot collisions from happening [7, 8].

Traditional systems rely on radar, ultrasonic or lidar sensors for distance estimation; while effective at doing this, their cost increases while offering limited information about object types. Camera-based methods combined with deep learning offer more flexible approaches by enabling detection, classification and

© The Author(s) 2026
S. Sharma et al. (eds.), *Proceedings of the International Conference on Recent Advances in Intelligent and Sustainable Technologies (RAIST 2026)*, Atlantis Highlights in Intelligent Systems 17,
https://doi.org/10.2991/978-94-6239-707-1_20

localization using affordable hardware; advances in convolutional neural networks (CNN) have further increased performance under various lighting and traffic conditions [6, 3].

The YOLO family of models is widely utilized for real-time detection due to its balance between speed and accuracy [2, 1]. However, full-scale models may not always be suitable for low-power devices [12, 9]. Lightweight variants like YOLOv4-Tiny can reduce computational requirements while still offering reasonable performance making them suitable for embedded platforms [12, 9].

Recent studies have integrated detection with region-based reasoning for blind spot monitoring [11, 10]; however, most solutions still rely on expensive hardware and don't adequately track real-time tracking on embedded devices like Raspberry Pi due to limited resources and different operating conditions [4, 5].

This work introduces a Raspberry Pi-based blind spot detection and tracking system designed to meet these challenges. Our approach employs a lightweight model with motion-based tracking and simple decision rules in order to create reliable alerts, in an attempt to strike an equilibrium between performance, cost and real-time operation in embedded ADAS applications.

2 Background and Related Work

Blind Spot Detection (BSD) is an integral component of Advanced Driver Assistance Systems (ADAS), designed to prevent accidents when vehicles outside a driver's field of vision need to change lanes during lane changes. Early BSD systems utilized radar and ultrasonic sensors that provided distance measurements but limited information regarding object types - leading them to generate unnecessary alerts in complex traffic scenarios[8, 7].

Deep learning's rapid advances have propelled camera-based approaches into prominence. Convolutional neural networks (CNNs) enable simultaneous object detection and classification for greater contextual understanding than traditional sensor methods; making vision-based systems more adaptable to changing traffic and environmental conditions[6, 3].

Real-time detection often uses YOLO models due to their balance between speed and accuracy[2]; however, their standard versions can often be too heavy for embedded devices; lighter variants like YOLOv4-Tiny can help reduce computational requirements while still offering acceptable performance - making them suitable for edge platforms[12, 9].

Recent studies have combined object detection and region-of-interest (ROI) strategies to increase blind spot monitoring [11, 10]. Although promising results were achieved, many approaches either relied heavily on powerful hardware or only focused on detection accuracy with limited consideration for tracking consistency and embedded deployment - running such systems on devices like the Raspberry Pi presented additional challenges related to limited resources and variable operating conditions [4, 5].

Table 1. Comparison of Recent Blind Spot Detection Approaches

Ref.	Sensor	Model	Platform	Limitation
Chang et al. [7]	Camera	YOLO-based CNN	Desktop	Not optimized for embedded use
Lee and Park [11]	Rear camera	CNN	Embedded hardware	Higher computational cost
Jiang et al. [10]	Camera	Deep CNN	Automotive-grade system	No explicit tracking analysis
Boddu et al. [12]	Camera	YOLOv4-Tiny	Raspberry Pi	Detection-focused, no BSD logic
This work	Camera	YOLOv4-Tiny + tracking	Raspberry Pi	Low-cost real-time BSD

As much as detection techniques have advanced, low-cost hardware systems that integrate detection, tracking and real-time operation are still required for real ADAS applications. In this work, a lightweight BSD framework was created to address this need by combining object detection with motion-based tracking on a Raspberry Pi for providing an economical and practical ADAS solution for real world ADAS applications.

2.1 Comparison of Related Works

Table 1 provides a comparison of recent blind spot detection and vision-based approaches, covering aspects such as sensing methods, model selection, hardware platforms and their respective limitations. A closer examination reveals that many proposed systems either cater to high-end automotive environments or focus on increasing detection accuracy alone - failing to fully address essential features like consistent real-time tracking or compatibility with low-power embedded devices; This indicates there remains a need for solutions capable of operating efficiently while remaining affordable and resource-constrained platforms.

3 Methodology

This section describes our real-time blind spot detection and tracking system on a Raspberry Pi 3B+, using a lightweight deep learning model combined with motion-based tracking to achieve reliable performance with limited computational resources. Simple persistence-based rules are employed to improve reliability while decreasing false alerts while maintaining real-time operation on low-cost embedded hardware.

Figure 1 illustrates the overall system design. A camera captures images of blind spot regions continuously and streams them directly to a Raspberry Pi 3B+ for processing, including detection, tracking and risk evaluation locally without cloud support.

3.1 System Architecture

After assessing conditions, the system generates alerts using an LCD display, an LED indicator, and a buzzer. Following a modular approach where perception, tracking, and alert generation are separated components; this ensures smooth operation while also reducing delays for real-time ADAS applications.

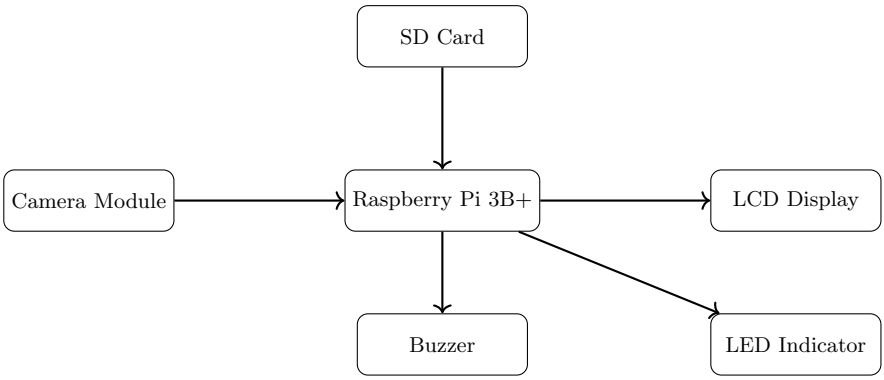

Fig. 1. System architecture of the proposed real-time blind spot detection and tracking framework using Raspberry Pi 3B+.

3.2 Hardware Components

The proposed system comprises the following hardware components:

- **Raspberry Pi 3B+:** Serves as the central processing unit responsible for video acquisition, deep learning inference, object tracking and peripheral control. With its ideal balance between computational capability, energy consumption and cost makes this unit suitable for embedded ADAS prototyping and deployment. [4].
- **Camera Module:** Record continuous video streams of the blind spot region and provide visual data input into the perception pipeline.
- **SD Card:** Storage area for an operating system, trained neural network weights and application software.
- **LCD Display:** Presents real-time visual feedback regarding system status and blind spot warnings.
- **LED Indicator:** Blind Spot Alert provides instantaneous visual warning for detected blind spot hazards.
- **Buzzer:** Generate audible warnings during critical events to heighten driver awareness and alertness.

3.3 Object Detection and Tracking

Object detection is performed using a lightweight YOLO model that processes images in one pass to achieve low latency suitable for ADAS applications [2, 1]. In particular, YOLOv4-Tiny is selected due to its lower computational requirements and suitability for embedded devices [12].

Filtered detections are filtered using confidence thresholds and spatial constraints that define the blind spot region. To maintain stability over time, centroid-based tracking method associates objects across frames based on position and motion; this helps preserve identity during brief occlusions or missed detections[5]; providing reliable estimation of movement within the monitored area.

Algorithm 1 Real-Time Blind Spot Object Detection and Tracking

Input: Video stream from camera module
Output: Blind spot warning signals (LCD, LED, Buzzer)
Initialize Raspberry Pi peripherals (camera, LCD, LED, buzzer)
Load trained YOLOv4-Tiny model and class labels
Set confidence threshold τ_c and blind spot region R_{bs}
while system is active **do**
  Capture video frame F_t from camera
  Preprocess F_t (resize, normalization)
  Perform object detection on F_t using YOLOv4-Tiny
  Obtain set of detections $D_t = \{(b_i, c_i, s_i)\}$
  for each detection $d_i \in D_t$ **do**
    if $s_i < \tau_c$ **then**
      Discard d_i
    else
      Compute centroid p_i of bounding box b_i
      Assign d_i to an existing track or create new track
    end if
  end for
  Update object tracks using centroid distance and motion history
  Remove tracks not observed for N consecutive frames
  for each active track T_j **do**
    if $p_j \in R_{bs}$ **then**
      Mark object as blind spot candidate
    end if
  end for
  if blind spot candidate persists for K frames **then**
    Activate LED indicator
    Display warning on LCD
    Trigger buzzer alert
  else
    Deactivate warning signals
  end if
end while

3.4 Alert Generation and Decision Logic

Dangers can be identified using both spatial and temporal criteria. A vehicle will be considered a threat only if it enters and stays within its blind spot area for an extended period, thus eliminating false alarms from temporary detections.

Once confirmed, warnings are generated through LCD, LED and buzzer to provide visual and audio feedback to improve driver awareness in ADAS scenarios [7, 6]. Alerts will automatically go away once an object leaves its region or no longer meets persistence criteria.

3.5 Blind Spot Detection and Tracking Algorithm

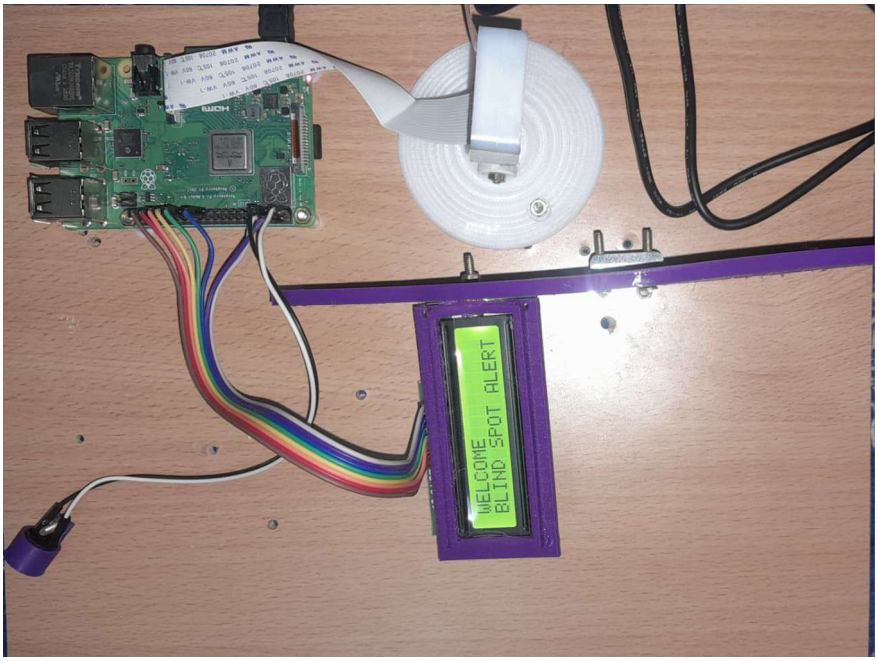

Fig. 2. Experimental setup of the proposed blind spot detection system implemented on Raspberry Pi 3B+.

Algorithm 1 outlines the overall process of this proposed method. The system initializes all required components and loads the YOLOv4-Tiny model with predefined parameters; with each new frame arriving, objects are detected and tracked using a centroid-based approach for tracking purposes.

Objects that remain within the blind spot region for an extended period are treated as hazards and trigger visual and audio alarms, while tracks not observed for several frames are deleted to preserve efficiency. This approach balances accuracy, stability, and real-time performance on resource-limited embedded hardware platforms.

4 Experimental Setup and Dataset Description

This section details the experimental setup and datasets to assess our real-time blind spot detection and tracking system. These experiments seek to investigate how well it detects objects reliably over time while maintaining tracking, and whether it can operate efficiently on low-cost embedded platforms in real time.

4.1 Experimental Setup

Fig. 2 depicts our experimental setup, consisting of a Raspberry Pi 3B+ equipped with a camera placed to monitor the side-rear region corresponding to blind spots

Table 2. Summary of datasets used for experimental evaluation

Dataset	Src.	Scenes	Obj. classes	Purpose
Public dataset	Public	Urban, highway	Car, truck, motorcycle, pedestrian	Detection evaluation
Self-collected	Real	Urban, suburban	Car, motorcycle, truck	Real-time BSD, tracking

on vehicles, thus capturing vehicles in adjacent lanes that may otherwise not be visible to drivers.

All video processing takes place locally on a Raspberry Pi without cloud support, running a lightweight Linux system to perform detection, tracking and decision making as described in Section 3.1. A YOLOv4-Tiny model stored on SD card is used for real-time inference.

Once an object or action is detected, an LCD, LED, and buzzer provide alerts with both visual and audio feedback, making this system suitable for low-cost ADAS deployment.

4.2 Dataset Description

Evaluation utilizes both public driving datasets and videos recorded with the prototype to allow testing under both standard benchmarks and real-world conditions.

Public datasets allow researchers to assess performance across various traffic scenarios, road types and object classes such as cars, trucks, motorcycles and pedestrians. Video footage recorded from urban and suburban environments presents unique tracking challenges such as lane changes, lighting variations, motion blur and occlusions that need to be evaluated consistently for accurate tracking evaluation.

Table 2 presents an overview of datasets including source, scene type, object classes and evaluation purposes. Leveraging both benchmarked data as well as real world information allows a clearer understanding of system performance and real time behavior on embedded hardware systems.

Blind spot regions are defined using the camera's field of view and vehicle alignment, with only objects identified within this region considered for hazard evaluation. Ground truth for recorded sequences can either be manually verified at frame level, or obtained through dataset annotations and manually reviewed dataset annotations.

4.3 Evaluation Protocol

System performance is evaluated based on detection accuracy, tracking consistency and real-time operation. Accuracy is measured using precision and recall; tracking quality can be assessed by maintaining object identities across frames; while real-time performance can be determined using Raspberry Pi processing speed averages.

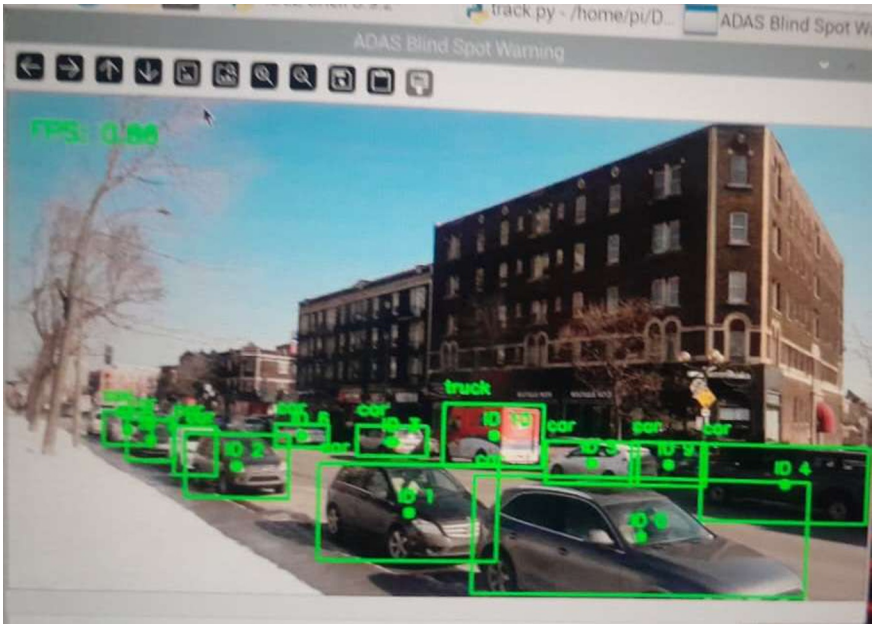

Fig. 3. Example frames showing real-time detection and tracking of adjacent vehicles using YOLOv4-Tiny.

To maintain consistency across experiments, all use the same model, confidence thresholds, and tracking parameters. This enables fair comparison across datasets and real-world scenarios and shows whether or not the system is suitable for embedded blind spot monitoring.

5 Results and Discussion

This section presents the results from testing the blind spot detection and tracking system on a Raspberry Pi 3B+. Our analysis focused on how well it performed in terms of accuracy, tracking stability, alert response time, as well as its ability to operate under real world conditions.

5.1 Detection and Tracking Behavior

Example results can be seen in Fig. 3 The system successfully detects nearby vehicles such as cars and trucks in its field of view while providing centroid-based tracking to maintain consistent identities for these objects across frames resulting in seamless motion tracking.

As vehicles move into the blind spot from adjacent lanes, the system continues to track them despite brief occlusions or changes in speed. Occasional detection gaps caused by motion blur or partial obstruction are handled effectively by

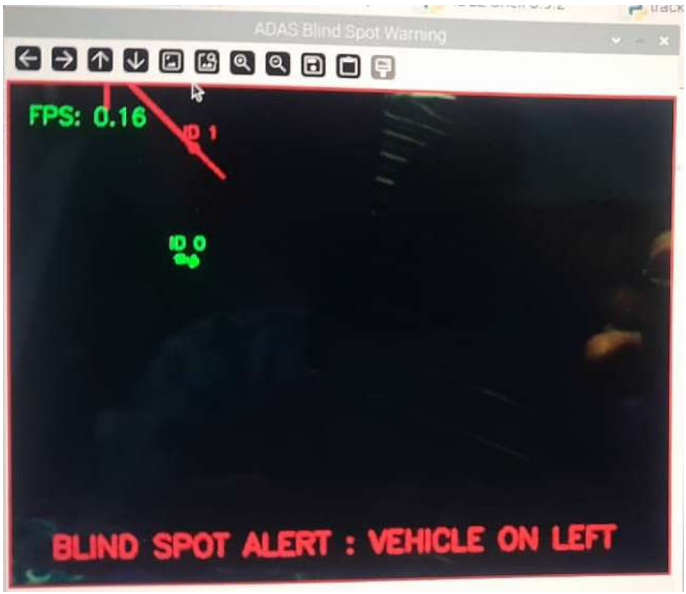

Fig. 4. Software-generated blind spot warning displayed when a vehicle remains in the monitored region.

the tracking mechanism and prevent sudden disruptions of output. Overall, this behavior indicates greater robustness compared to approaches that rely solely on frame-by-frame detection.

5.2 Blind Spot Alert Visualization

Fig. 4 demonstrates how real-time alerts are created and displayed during real-time operation. The system issues an alert only if an object stays within its defined blind spot region for an extended period, helping avoid unnecessary alerts that would arise from short or momentary detections. Once an object leaves this region, any alert issued immediately clears as evidence of quick scene changes being addressed by the system.

5.3 Embedded Alert Output

Fig. 5 shows the hardware alert system. When vehicles enter into blind spots, an LCD displays a warning while LED and buzzer activate to create visual and audible signals to increase driver awareness in various conditions and in conjunction with Perception Module.

5.4 Quantitative Performance Analysis

Evaluation was performed using public datasets and manually verified recorded frames, with vehicles like cars, trucks, and motorcycles as the focus. The system

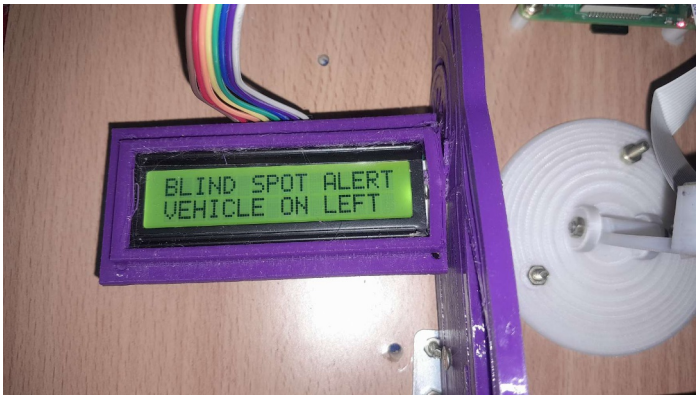

Fig. 5. Blind spot warning displayed on the LCD module during experimental testing.

achieved an accuracy range between 85-98% with controlled false alert rates at around 83% for detection while providing real-time driver assistance services.

Runtime performance for the Raspberry Pi 3B+ was approximately 12–15 frames per second depending on scene complexity, making this speed sufficient for timely alerts without needing additional hardware acceleration.

5.5 Discussion

Results demonstrate that effective blind spot monitoring is possible using a lightweight model on lower-cost hardware. While not comparable with high-end systems, our approach provides a practical compromise between accuracy and computational efficiency for affordable vision-based ADAS solutions.

6 Conclusion and Future Work

This work presents a real-time blind spot detection and tracking system designed for use on a Raspberry Pi 3B+. Utilizing a lightweight YOLOv4-Tiny model with motion-based tracking and simple region rules, reliable monitoring under limited resources while still maintaining stable real-time performance without additional hardware acceleration is achieved.

Testing under realistic driving conditions demonstrated consistent detection, tracking and alert generation; LCD, LED and buzzer alerts working together seamlessly for increased driver awareness. These results underscore that camera-based solutions offer cost-effective alternatives to sensor-heavy ADAS systems.

Future work will focus on improving performance under low-light conditions, adverse weather and higher speeds by including features like advanced tracking

methods, sensor fusion with radar or ultrasonic modules, higher resolution inputs and optional hardware acceleration; as well as adaptive blind spot modeling tailored specifically for different vehicle types.

References

1. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020). doi:10.48550/arXiv.2004.10934
2. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2020). doi:10.48550/arXiv.1804.02767
3. Saeed, A., Al-Hourani, A., Jamalipour, A.: A survey on machine learning techniques for intelligent transportation systems. IEEE Communications Surveys & Tutorials 23(3), 1443–1476 (2021). doi:10.1109/COMST.2021.3074128
4. Chen, L., Zhang, H., Li, Y.: Embedded vision systems for intelligent vehicles: A review. Sensors 21(11), 3703 (2021). doi:10.3390/s21113703
5. Huang, Z., Wang, X., Liu, J.: Lightweight object tracking for embedded vision applications. IEEE Access 10, 21456–21467 (2022). doi:10.1109/ACCESS.2022.3149256
6. Zhao, D., Peng, H., Sun, J.: Vision-based perception for advanced driver assistance systems: A review. IEEE Transactions on Intelligent Transportation Systems 23(8), 12015–12029 (2022). doi:10.1109/TITS.2022.3143621
7. Chang, Y., Lin, C., Hsu, T.: Vision-based blind spot detection for intelligent vehicles. IEEE Sensors Journal 23(6), 6124–6135 (2023). doi:10.1109/JSEN.2023.3241879
8. World Health Organization: Global status report on road safety 2018. World Health Organization, Geneva (2018). https://www.who.int/publications/i/item/9789241565684
9. Wang, R., Liu, S., Chen, Q.: Edge AI for real-time ADAS applications using lightweight deep learning models. IEEE Internet of Things Journal 10(4), 3312–3324 (2023). doi:10.1109/JIOT.2023.3239911
10. Jiang, M., Li, X., Zhou, Y.: Vision-based blind spot monitoring using deep neural networks. IEEE Access 12, 45671–45682 (2024). doi:10.1109/ACCESS.2024.3372159
11. Lee, J., Park, S.: Camera-based blind spot detection and lane change assistance using deep learning. Sensors 25(2), 812 (2025). doi:10.3390/s25020812
12. Boddu, R., Kumar, S., Reddy, M.: Real-time object detection on edge devices using YOLOv4-Tiny. Journal of Real-Time Image Processing 22, 45–58 (2025). doi:10.1007/s11554-024-01489-3

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.