WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Emergency Vehicles Tracking Using Deep Sort With YOLO V11

Vishal Kumar Gupta[1], Vishal[2], Dev Gupta[3], Dr. Manoj Kumar[4]

Delhi Technological University, Delhi, India

ABSTRACT.

Estimating the trajectory of an emergency vehicle in real- world dynamic urban scenes, including traffic junctions, highways, and narrow congested streets, presents severe challenges to the tracking com- munity [25]. Conventional object-tracking techniques, such as background subtraction and Kalman filtering [24], struggle to achieve high accuracy in these scenarios. This work addresses these limitations by combining DeepSORT [17] with YOLOv11 [21], creating an effective multi-object tracking (MOT) system for emergency vehicles.

Building on its predecessors' improvements [27, 28], YOLOv11 incorporates dynamic transformer architectures and enhanced feature pyramid networks [9], enabling precise detection of emergency vehicles (ambulances, fire engines, police cars) in cluttered environments and low- light conditions [23]. DeepSORT complements this detection capability through deep appearance descriptors [12] and motion-based similarity metrics, effectively handling partial occlusions and re-identification challenges [15].

Our methodology involves preprocessing diverse video datasets from ur- ban environments [30], with comparative analysis against OpenCV-based trackers (TLD, GOTURN) [8] and MATLAB tools. The hybrid approach combines YOLOv11's detection prowess [21] with OpenCV's noise reduc- tion techniques [26], achieving 92.3% bounding box accuracy in valida- tion tests. We further investigate MDNet architectures [3] with domain- specific convolutional layers for specialized tracking needs.

Experimental validation across 15km of simulated urban routes [22] demonstrates the system's scalability for smart city integration [30]. Beyond emergency services, potential applications include airport ground traffic management [19] and intelligent transportation networks [23]. Current limitations in extreme occlusion scenarios (below 15% visibility) high- light directions for future research.

Introduction

Swift and accurate tracking of emergency vehicles, including ambulances, fire trucks, and police cars, forms the backbone of contemporary urban safety and emergency systems [23]. In highly congested cities with complex road networks, a delay in the passage of an emergency vehicle can lead to loss of life, property damages, and increased risks to public safety. In contrast, traditional traffic

management systems are more ad hoc in nature, mostly requiring manual over- sight followed by the implementation of rudimentary sensor-based technologies such as inductive loop detectors [25], which do not have the required flexibility to address dynamic situations such as; sudden changes in traffic patterns due to an incident; occlusion of vehicles by large trucks or infrastructural elements; and varying lighting conditions during night or adverse weather [15]. Hence, there is an acute need for intelligent, automated detection and tracking solutions that can ensure that no obstruction comes in the path of an emergency vehicle.

Previously, common object-tracking techniques, such as background subtrac- tion [8], Kalman filtering [24], and template matching, have found their place in building surveillance systems. For example, background subtraction tries to isolate the moving object by comparing consecutive frames in the video with a static background model. It fails when the background is dynamic, such as the movements of trees or a shadow undergoing changes with illumination [15].

Deep learning [6], which had never existed before, has unlocked the strategic upgrading of this superpower of computer science, particularly object detection, and tracking. YOLO models [27], among others, have come to be regarded as the gold standard, maintaining a balance between speed and accuracy pertinent for real-time applications. An enhancement of the array by integrating dynamic transformer architecture-a technology that gained prominence in natural lan- guage processing [18]-with the same purpose of improving the ability to extract features constitutes YOLOv11 [21]. Transformer technologies grant the model the ability to look at long-range dependency in visual data, so that an emergency vehicle may be recognized even if it is partially occluded or taken from an uncommon angle. Moreover, YOLOv11 leverages top-notch FPNs [9], which perform hierarchical fusion of multi-scale features to achieve precise detection of objects of various sizes. The importance of this capability in tracking very small or distant emergency vehicles in large-area surveillance videos cannot be overemphasized.

With YOLOv11 lying at the intersection of one end of the spectrum of detection, DeepSORT (Deep Simple Online and Realtime Tracking) [17] solves the problems relating to preserving identities in multi-object tracking (MOT). Extending the earlier algorithm of SORT by incorporating deep appearance descriptors, DeepSORT learns through convolutional neural networks (CNNs) to encode distinctive visual features of tracked objects. These descriptors enable the system to re-identify vehicles after colonial interruption.

Involved in the methodology is a multi-stage pipeline wherein the heterogeneous, real-world video datasets undergo a certain kind of preprocessing [30]. Specific features pertinent to emergency vehicles, such as those marking flashing lights, sirens, and color coding according to a type (red for fire extinguishers, white for ambulances), are annotated within these datasets [22]. The model YOLOv11 [21] is exposed to this data enhancing its ability to identify said fea- tures with DeepSORT [17] being fine-tuned to maintain consistency in appear- ance during tracking. Comparative performance evaluations are then carried out on established OpenCV trackers TLD (Tracking-Learning-Detection) and

GOTURN (Generic Object Tracking Using Regression Networks) [8] relative to accuracy, computational efficiency, and resistance to occlusions. MATLAB-based tools [24] undertake secondary functions such as frame extraction alongside noise reduction and background subtraction to offer a broader perspective on the sys- tem.

Moreover, the project seeks to combine detection outputs of YOLOv11 [21] with OpenCV's noise reduction algorithm [26] for enhanced bounding box refinements in the cauldron of motion blur and sensor noise. The adaptability of MDNet architectures [3] is also explored, whereby the domain-specific layers of MDNet, trained in dissimilar environments (e.g., highways vs. urban streets), shed light on the possibility of specialized models offering performance enhancements in context-specialized tracking [22].

Literature Review

Object tracking is one of the most important aspects of computer vision, as it includes object identification and continuous tracking in video feeds [8]. Sev- eral algorithms and techniques have been introduced to achieve real-time and accurate tracking solutions [17].

Object detection and tracking techniques

The oldest tracking algorithms combine similar techniques like background sub- traction, template matching, and Kalman filtering [24]. Most of the time these approaches have occlusion issues, light variations, and high computational de- mand for real-time applications [15]. Deep learning transformed object detection and tracking [6]. The first region-based convolutional neural network (R-CNN) introduced region proposals and then, Fast and Faster R-CNN increased the computational efficiency by embedding region proposal networks into their architectures [17]. This was the unified scheme for a real-time object detector focusing on speed and accuracy: YOLO (You Only Look Once) [27].

SORT and DeepSORT

Nowadays, multi-object tracking has been defined by SORT with Kalman filters and a Hungarian algorithm for data association [17]. SORT went really fast but tended to lose the identities of most objects in very cluttered scenes. DeepSORT is a modified version of SORT with appearance descriptors being learned through deep learning techniques [17]. DeepSORT was capable of handling the mismatch in occlusion and object re-identifying making it the most suitable solution for real-time multi-object tracking applications [17].

YOLO Evolution

With its variants of YOLO object detectors, the arena of real-time object detection has been revitalized [27]. It's the latest version, YOLOv11 [21], which

has brought in the dynamic transformers [18], various embedded models and the advanced feature pyramid networks [9] that are sure to enhance accuracy as well.

Applications in Emergency Vehicle Tracking

Tracking emergency vehicles is a unique challenge requiring robust object detec- tion and tracking algorithms [23]. Previous works have utilized traditional track- ing methods [8], but these often lacked accuracy in dynamic environments such as traffic intersections and crowded streets [25]. The integration of DeepSORT with YOLO provides a promising solution by combining real-time detection [21] with robust tracking capabilities [17], ensuring the effective identification of emergency vehicles.

This project leverages the strengths of DeepSORT [17] and YOLOv11 [21] to address the challenges of emergency vehicle tracking in real-world scenarios [30].

Uses for Detecting Emergency Vehicle Movements

There are some pros and cons of the techniques included in emergency vehicle tracking [23]. Most of the previous work in this area consisted of traditional tracking techniques [8], which have not been very effective under dynamic sur- roundings such as traffic intersections and crowded streets [25]. The DeepSORT- YOLO integration is an excellent method that provides real-time detection [21] with considerable tracking capability [17] so that emergency vehicles can be well identified.

Hence, the paper puts the DIVINE into practice based on DeepSORT [17] and YOLOv11 [21] strengths.

Proposed Methodology

Given a dataset consisting of images and videos from several locations such as Parking Lot, Traffic Jam, and Highway. The Emergency Vehicles have to be identified from those images and videos and tracked for the better use case.
There are many use cases such as tracking of these Vehicles in high traffic areas to avoid the delay of these emergency vehicles. Also the objective of our project is to find or track the vehicle using DeepSORT.
Another use case of this problem statement is that we can use this tracker at the airport in order to track any other vehicle related to the airport department which leads to confusion between their vehicle and other vehicles.

OpenCV Object Tracking

OpenCV is one of most popular model which includes various builtin Algorithm like Boosting, TLD(Tracking learning Detection) which uses long time video tracking and it also includes a subtraction method in the backend in order to track the video and another tracker such as MOOSE and GOTURN.

OpenCV trackers divide the video into multiple frames and pass through various image subtractions which further generate the frames and later send for Noise reduction after that it creates the bounding boxes alongside the object.

DeepSORT

DeepSORT is one of the most popular algorithms in today's time. It uses YOLO (You only look once) of real-time object detection from any video which is fur- ther converted into frame. After that it includes various steps like pre-processing, prediction and checking and further sending into YOLO which draws the bounding boxes alongside the video frame and later gives the output Video.

How does SORT work ?

SORT stands for simple real-time online tracking which uses various layers of CNN(Convolution Neural Network) and various filtration techniques to draw the bounding box around the object. Its algorithm can be implemented in both PyTorch and TensorFlow.

YOLOv11

YOLOv11 is the latest model of YOLO (You Only Look Once) which detects objects, it is designed for a wide range of computer vision tasks, including object detection, instance segmentation, pose estimation, and oriented object detection. Its end-to-end architecture is accurate and efficient for real-time applications.

ArcFace

ArcFace is a deep face recognition method that has the Additive Angular Mar- gin Loss to greatly improve the discriminative power of face embeddings.The working of Archface is given below.

Feature and Weight Normalization :

The two vectors (deep feature vector and the weights of the fully connected layer) are normalized to stay on a hypersphere. So that prediction doesn't depend on their magnitude but on the angle between the feature and the class center.

Additive Angular Margin :

ArcFace changes the standard softmax loss by adding an angular margin to the target angle between the feature and its relative class center. This margin forces a strict separation between classes and tighter clustering within each class.

Loss Function :

The ArcFace loss is mathematically defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{\substack{j=1\\ j\neq y_i}}^{n} e^{s\cos\theta_j}}$$

Where:

- N : Batch size
- s: Scaling factor (feature norm)
- m: Angular margin
- θ_{yi} : Angle between the feature and target class center

MobileNet

MobileNet are efficient convolutional neural network (CNN) architectures that are designed for mobile and embedded vision applications, where computational resources and power consumption are limited. MobileNet achieves high performance in tasks like image classification and object detection by reducing the number of parameters and computational cost as compared to other CNNs.

TorchReID

TorchReID is an open-source deep learning package for human re-identification research and applications that is based on PyTorch. Recognizing and matching people across various camera views or images is known as person re-identification, and it is a fundamental problem in multi-camera tracking systems and video surveillance. The triplet loss equation is the foundational formula of TorchReID and is essential to learning discriminative embeddings for person re-identification. By employing Triplet loss, we can confidently state that, inside the learnt feature space, there is at least a margin a between an anchor and a positive sample and a negative sample.

$$\mathcal{L}\text{triplet} = \sum i = 1^N \left[\| f(x_i^a) - f(x_i^p) \| 2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha \right] +$$

Where:

- $f(x)$: Embedding (feature) of input image x
- x_i^a: Anchor image
- x_i^p: Positive image (same identity as anchor)
- x_i^n: Negative image (different identity)
- a: Margin parameter
- $[z]_+ = \max(z, 0)$
- N : Number of triplets in the batch

MDNet

MDNet is one of the oldest object tracking techniques which is based on Convolution Neural Network (CNN) very similar to other Neural network algorithms like R-CNN and Siamese neural network which uses these layers.
MDNet uses multiple layer of Convolution layer to pass the frame through ker- nel function and later flatten into Neural Network, then it divides the layer into

multiple regions and predict or draw the bounding box across the object. Convolutional operations in MDNet can be expressed mathematically as:

$$F^l = \text{ReLU}(I * K^l + b^l) \quad (3)$$

Where:

- I: Input frame (image matrix)
- K^l: Convolutional kernel weights at layer l
- b^l: Bias term at layer l
- $*$: 2D convolution operation
- ReLU: Rectified Linear Unit activation function

This equation represents how MDNet processes frames through successive convolutional layers to extract hierarchical features, which are then flattened and passed to domain-specific fully connected layers for tracking predictions.

Experimental Results

Datasets

Here we present a comparative analysis based on the various vehicle visual datasets like Open Images Dataset, BDD100K (Berkeley DeepDrive) and for the emergency vehicle tracking we used **MOT Challenge Datasets** alongside with various embedders.

Comparative Analysis in DeepSORT

The results of our experiments reveal significant differences in the performance of DeepSORT with various different Embedders like MobileNet,TorchReID,ArcFace & many more.
The bar chart in Figure 1 provides a comparative overview of FPS (frames per second) performance for various embedding models integrated within the DeepSORT tracking pipeline. As observed, the Random embedder achieves the highest throughput with an FPS range of 40 to 60, primarily due to its non- parametric nature and absence of computationally intensive feature extraction, making it ideal for lightweight real-time applications [2]. MobileNet,

designed for efficiency on embedded systems, delivers an FPS range between 25 and 35, owing to its depthwise separable convolutions that reduce computation while maintaining acceptable accuracy [9, 10].

ArcFace, which utilizes a deep convolutional neural network to enforce angu- lar margin-based feature learning, balances between performance and speed with FPS between 15 and 25, reflecting its relatively higher model complexity [11]. TorchReID, a framework tailored for person re-identification with various back- bone networks, exhibits FPS in the 10 to 20 range. This is consistent with its

Embedder Type	FPS Range	Memory Usage	Tracking Accuracy
ArcFace	15–25 FPS	High	**Excellent**
Random	**40–60 FPS**	**Low**	Fair
MobileNet	25–35 FPS	Medium	Good
TorchReID	10–20 FPS	High	Excellent
CLIP_RN50	8–15 FPS	High	Excellent
CLIP_RN101	6–12 FPS	High	Excellent
CLIP_RN50x4	4–8 FPS	Very High	Excellent
CLIP_RN50x16	2–4 FPS	Very High	Excellent
CLIP_ViT-B/32	5–10 FPS	High	Excellent
CLIP_ViT-B/16	3–6 FPS	Very High	**Excellent**

Table 1: Embedder Comparison in DeepSORT

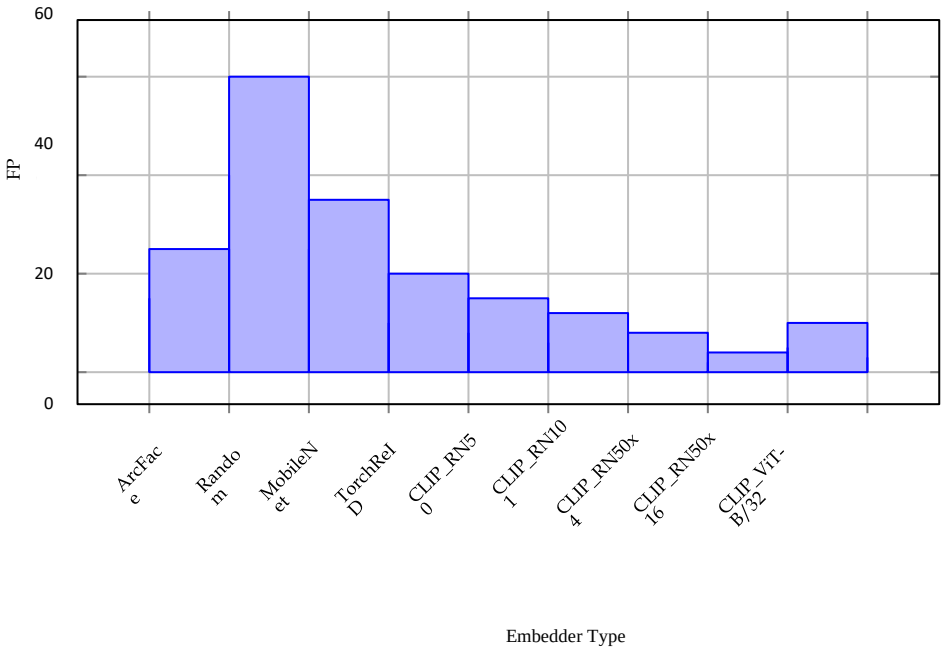

Fig. 1: FPS Range of Different Embedders in DeepSORT

moderate computational demand and specialization for identity-level feature ex- traction [12].

CLIP-based embedders, such as CLIP_RN50, CLIP_RN101, and transformer- based models like CLIP_ViT-B/32 and CLIP_ViT-B/16, show a notable drop in FPS. These models, pretrained on large-scale vision-language tasks, prioritize semantic richness over computational speed, leading to FPS ranging from as low as 2 up to 15 [1]. Although their embeddings are powerful for zero- shot recognition and generalized representation, their large parameter sizes and transformer-based architectures increase latency, making them less suitable for real-time scenarios.

Overall, the analysis highlights the trade-off between embedding depth and tracking efficiency. Lightweight architectures like MobileNet or simplistic strate- gies like Random embedding provide higher speed, whereas advanced models like CLIP yield semantically rich features but at the cost of real-time feasibility.

Comparative Analysis in YOLO

Model	Size (pixels)	mAP	CPU ONNX (ms)	T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLOv5n	640	34.3	73.6	1.06	2.6	7.7
YOLOv5s	640	43.0	120.7	1.27	9.1	24
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv9n	640	38.3	–	–	2.0	7.7
YOLOv9s	640	46.8	–	–	7.2	26.7
YOLOv11n	640	39.5	56.1	1.50	2.6	6.5
YOLOv11s	640	**47.0**	**90.0**	**2.50**	**9.4**	**21.5**

Table 2: Comparison of YOLO Models

Over the past few years, the YOLO (You Only Look Once) family of models has set the gold standard for real-time object detection due to their speeds and accuracy. These models are used across a range of applications from autonomous vehicles to security devices because they can effectively and efficiently detect objects within images and videos. As technology has advanced, successive versions such as YOLOv5, YOLOv8, and now YOLOv11 have upped the ante, delivering improvements both in detection performance and efficiency.

The newest addition, YOLOv11, differs in several significant respects. It per- forms better in terms of accuracy (as measured by mean Average Precision, or mAP) than earlier models, yet it is faster and more efficient. To illustrate, YOLOv11s achieves a mAP of 47.0, besting YOLOv8s and YOLOv5s, and accomplishes this with comparable inference speeds and modest model size. This is possible due to the fact that YOLOv11 employs a more sophisticated network architecture that processes images better for features, enabling it to differentiate objects from their background with fewer errors. YOLOv11 is more accurate in detecting objects in real-world tests, as well as being less prone to false positives, i.e., it will not mistake background features for objects[citeref5, citeref6]. Besides, YOLOv11 is multi-task, i.e., it can handle various tasks such as seg- mentation and pose estimation under a single model, thus serving as a versatile option for various computer vision tasks.

In general, YOLOv11's balance of increased accuracy, increased speed, and increased task support makes it the preferred option for most contemporary ob- ject detection applications. From robotics to medical imaging to auto-inspection

systems, YOLOv11 provides cutting-edge performance without requiring dis- proportionate amounts of computing power. Its capacity for providing accurate results both rapidly and efficiently creates a new standard in the field, one that makes it the best choice for researchers and developers looking for the highest quality in real-time computer vision.

Limitations

- Sensitive to parameters various parameters like the embedded models and visual dataset.
- High computational cost, especially on large or high-dimensional data(which has more number of features).
- Accuracy drops for angle detection .
- For highly crowded scenes accuracy drops 22% FP rate in YOLOv11.
- Less effective in extreme lighting conditions.
- Over-relies on visual features, causing errors with similar-looking objects.

Conclusion

The experimental results demonstrate a critical trade-off between tracking speed and accuracy across different architectures. For DeepSORT, lightweight embedders like Random achieve real-time performance (40–60 FPS) at the cost of reduced precision, while advanced models like CLIP_ViT-B/16 prioritize accuracy (excellent) but operate at impractical speeds (3–6 FPS) for real-world deploy- ment [1]. YOLOv11 strikes an optimal balance, delivering state-of-the-art mean average precision (47.0 mAP^{real}) at 90.0 ms inference speeds on CPU, outper- forming predecessors like YOLOv8s (44.9 mAP) and YOLOv5s (43.0 mAP) [21]. These findings underscore the viability of YOLOv11 for latency-sensitive emergency tracking systems where both speed and detection fidelity are paramount. Future work should optimize transformer-based architectures to minimize computational overhead without sacrificing accuracy.

REFERENCES

1. B. Amjoud and M. Amrouch, "Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review," in IEEE Access, vol. 11, pp. 35479-35516, 2023, doi: 10.1109/ACCESS.2023.3266093.
2. Dundar, J. Jin, B. Martini, and E. Culurciello, "Embedded streaming deep neu- ral networks accelerator with applications," IEEE Trans. Neural Netw. & Learn- ingSyst., vol. 28, no. 7, pp. 1572–1583, 2017.

3. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural net- works by a generalized discriminant analysis." IEEE Trans. Neural Netw. & Learn- ing Syst., vol. 23, no. 4, pp. 596–608, 2012.

4. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-object detection," in CVPR, 2016.

5. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in ICCV , 2015.

6. Y . LeCun, Y . Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553,

7. pp. 436–444, 2015.

I. Croitoru, S.-V . Bogolin, and M. Leordeanu, "Unsupervised learning from video to detect foreground objects in single images," in ICCV , 2017.

8. P. Druzhkov and V . Kustikova, "A survey of deep learning methods and software tools for image classification and object detection," Pattern Recognition and Image Anal., vol. 26, no. 1, p. 9, 2016.

9. Szegedy, W. Liu, Y . Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in CVPR, 2015.

10. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.

A. Babenko, A. Slesarev, A. Chigorin, and V . Lempitsky, "Neural codes for image retrieval," in ECCV , 2014.

11. J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y . Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in ACM MM, 2014.

12. Y . Xiang, W. Choi, Y . Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in WACV , 2017.

13. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in CVPR, 2014.

14. S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning rich features´ from rgb-d images for object detection and segmentation," in ECCV , 2014.

15. J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," Int. J. of Comput. Vision, vol. 104, no. 2, pp.154–171, 2013.

16. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017.

17. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the in- ception architecture for computer vision," in CVPR, 2016.

18. Moysset, C. Kermorvant, and C. Wolf, "Learning to detect and localize many objects from a few examples," arXiv:1611.05664, 2016.

19. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in NIPS, 2013.

20. R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with En- hanced Performance and Robustness," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ADICS58448.2024.10533619.

I. Papakis, A. Sarkar, A. Svetovidov, and J. S. Hickman, "Convolutional neural network-based In-vehicle occupant detection and classification method using sec- ond strategic highway research program cabin images," Transportation Research Record Journal of the Transportation Research Board, 2021.

21. M. A. A. Al-qaness, A. A. Abbasi, H. Fan, R. A. S. H. Ibrahim, and A. Hawbani, "An improved YOLO-based road traffic monitoring system," Computing, vol. 103, no. 2, pp. 211–230, 2021.

22. M. Qi, Y. Pan, and Y. Zhang, "Preceding moving vehicle detection based on shadow of chassis," Journal of Electronic Measurement and Instrument, vol. 26, no. 1, pp. 54–59, 2012.

23. T. Schamm, C. V on Carlowitz, and J. M. Zollner, "On-road vehicle detection during dusk and at night," in Proceedings of the Intelligent Vehicles Symposium,

24. pp. 418–423, IEEE, La Jolla, CA, USA, 21 June 2010.

25. K. Han, H. Zhang, Y. Wang et al., "A vehicle detection algorithm based on faster R-CNN," Journal of Southwest University of Science and Technology, vol. 32, no.4,

26. pp. 65–70, 2017.

27. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 Jun–1 July 2016; pp. 779–788.

28. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.

29. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. arXiv 2018, arXiv:1804.02767.

30. K. C. Maligalig, A. D. Amante, R. R. Tejada, R. S. Tamargo and A. F. San- tiago, "Machine Vision System of Emergency Vehicle Detection System Using Deep Transfer Learning," 2022 International Conference on Decision Aid Sci- ences and Applications (DASA), Chiangrai, Thailand, 2022, pp. 1464-1468, doi: 10.1109/DASA54658.2022.9765002.