International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 12 | Dec - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

# Intelligent Vehicle Detection and Velocity Prediction with Deep Neural Networks

**Ms.s Vanitha[1], Rajesh K[2], Dileep C[3] , Adarsha N[4]**

[1]*Assistant Professor, Computer Department & Presidency University, Bangalore*
[2]*PG Scholar Department of Computer Science & Presidency University, Bangalore*
[3]*PG Scholar Department of Computer Science & Presidency University, Bangalore*
[4]*PG Scholar Department of Computer Science & Presidency University, Bangalore*

---

**Abstract -** This paper proposes a highly efficient approach for estimating vehicle speeds using traffic surveillance video. The method enhances an existing state-of-the-art system that identifies 3D vehicle bounding boxes through vanishing point geometry combined with an object detection network. By substituting the original Retina Net detector with the more advanced YOLOv6 model, we achieve substantial improvements in processing speed while preserving the accuracy of speed estimation. Multiple versions of the proposed system are assessed based on vehicle detection performance and speed calculation accuracy. Comprehensive experiments conducted on different hardware platforms, including edge devices, reveal a marked increase in processing frame rate (FPS) compared to the previous state-of-the-art solution, with equal or better estimation precision. We further examine the balance between computational complexity and performance, demonstrating that compact models enhanced with post-training quantization provide the most effective solution for practical deployment scenarios. Using the Brno Comp Speed dataset, the optimized model surpasses earlier benchmarks in speed estimation accuracy, detection precision, and recall, while operating at nearly eight times higher speed. Additional testing on a separate dataset confirms the robustness and adaptability of the proposed approach across varying environments.

**Keywords** Vehicle speed estimation, Intelligent transportation systems, Edge computing, Visual traffic monitoring.

## 1.INTRODUTION

In the modern era of rapid urban growth and increasing mobility demands, intelligent transportation systems (ITS) play a vital role in managing complex traffic environments. Precise vehicle speed measurement is essential for traffic regulation, law enforcement, and the advancement of smart transport solutions. As a result, the application of computer vision and machine learning techniques for developing accurate and efficient traffic monitoring systems has gained considerable attention [1]. However, processing the large volumes of real-time video data generated by traffic cameras presents major computational challenges, since most vision-based surveillance approaches depend heavily on deep learning models. These challenges can be addressed through two primary strategies. One approach involves cloud-based processing, where powerful server-class hardware performs the computations. While effective, this method requires substantial bandwidth and sophisticated network infrastructure. Alternatively, edge computing enables data processing closer to the source, reducing network load and improving efficiency. This approach also enhances privacy by minimizing the need to transmit raw video data to centralized cloud servers [2]. In the context of vision-based vehicle speed estimation using edge computing, it is crucial to align algorithm design with the limited capabilities of edge devices. Therefore, this paper introduces enhancements to an existing state-of-the-art vehicle speed estimation framework [3], focusing on improving computational efficiency. The primary modification involves replacing the original RetinaNet [5] detector with the more efficient YOLOv6 [4] architecture as the base 2D detection model, which is adapted for detecting 3D bounding boxes. We train multiple versions of the vehicle detection model with different sizes and evaluate their performance across six hardware platforms, with particular emphasis on edge devices. Performance is measured in terms of speed estimation accuracy and computational cost using the BrnoCompSpeed dataset [6] and another benchmark dataset [7]. Additionally, we analyze the influence of model size, numerical precision, and input image resolution on both detection accuracy and processing efficiency. Our results indicate that although larger models provide more accurate 2D bounding box localization, this improvement does not necessarily enhance vehicle speed estimation accuracy, making smaller and more efficient models preferable for edge-based implementations. Experimental results on the Brno Comp

Speed dataset demonstrate that the optimized quantized model achieves lower mean (0.71 km/h vs. 0.75 km/h) and median (0.55 km/h vs. 0.58 km/h) speed estimation errors compared to the prior state-of-the-art approach [3]. It also outperforms previous methods in detection precision (91.63% vs. 87.74%) and recall (83.95% vs. 83.21%).Notably, the proposed model operates at nearly eight times the speed, making real-time deployment on edge devices practical, which was previously unattainable. Furthermore, several alternative lightweight configurations are presented, some achieving superior detection performance while maintaining comparable speed estimation accuracy. To support reproducibility and further research, the implementation and trained models are made publicly accessible.

## 2.Related works

Vision-based vehicle speed estimation typically involves several sequential stages, including traffic camera calibration, vehicle detection, and object tracking. This section reviews prior studies related to each of these components, as well as complete systems developed for vehicle speed estimation.

### 2.1 Traffic Camera Calibration

Traffic camera calibration is essential for obtaining precise real-world distance measurements on the road surface by determining the intrinsic and extrinsic parameters of the camera. Traditional calibration methods involve manual procedures using calibration patterns [8] or known physical measurements on the roadway [9]. Semi-automatic calibration techniques have also been proposed, combining automated processes with at least one known distance within the observed scene. These approaches frequently rely on vanishing point detection [10–14] or the analysis of parallel road features such as lane markings and curves [15], enabling more flexible and less labor-intensive calibration.

### 2.2 Computationally Efficient Object Detection

Object detection plays a critical role in vision-based vehicle speed estimation. Recent research has increasingly focused on improving computational efficiency to support real-time performance on resource- constrained systems. The YOLO family of detectors has emerged as a leading solution in this domain due to its balance between speed and accuracy [23–25]. YOLOv6 [4, 26], an evolution of earlier YOLO versions, is specifically designed for edge deployment by integrating re-parameterized network backbones and decoupled detection heads, resulting in improved efficiency on low-power hardware. In addition to YOLO-based solutions, other lightweight detection. Models [27, 28] have been developed to address scenarios with limited computational resources. Further performance optimization can be achieved through model compression techniques such as quantization and knowledge distillation. Quantization can be implemented either through Quantization-Aware Training (QAT) or Post-Training Quantization (PTQ). QAT introduces quantization during model training, enabling the network to adapt and reduce accuracy degradation, whereas PTQ applies quantization after training using calibration data to determine appropriate scaling and clipping parameters [29]. Knowledge distillation, on the other hand, improves the efficiency of smaller models by transferring knowledge from a larger, more complex model.

### 2.3Vision-Based Vehicle Speed Estimation

An alternative approach to traditional pipelines is presented in [45], where a deep neural network is employed to directly predict vehicle speeds from video input without requiring prior camera calibration. This method bypasses geometric calibration entirely, offering a fundamentally different strategy for speed estimation.

### 2.4Evaluation datasets

The evaluation of vision-based vehicle speed estimation techniques requires datasets that provide accurate ground truth vehicle speeds. In many cases, such ground truth values are obtained using external sensors such as induction loops embedded in the roadway, which enable precise speed measurement. Some datasets also include detailed annotations such as vehicle positions and license plate information, which further support comprehensive performance evaluation.

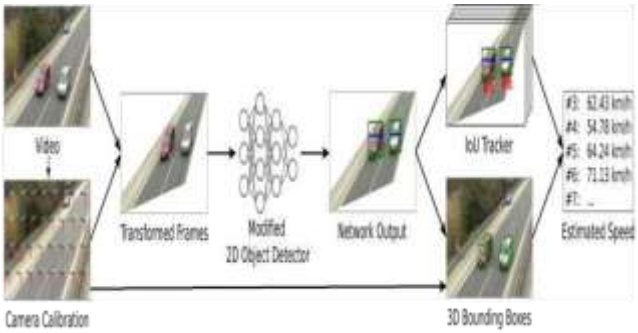



Fig.1 The vehicle speed estimation pipeline from [3]. During inference, the input images are first transformed using perspective transformation. This information is used to construct a 3D bounding box in the original image. The detected 3D bounding boxes are then tracked using a

Vehicle tracking plays a crucial role in associating detections of the same vehicle across consecutive frames. Commonly used tracking approaches include the Kalman filter [33], which serves as the basis for the SORT algorithm [34] and its extended variants [35–37], all of which track object bounding boxes over time. Given the improved reliability of modern object detectors, simpler tracking strategies such as Intersection-over-Union (IoU)-based tracking have also proven effective [38]. Alternatively, some methods perform detection and tracking simultaneously within a single deep learning framework [39, 40].

Vehicle speed is typically computed from the temporal progression of vehicle positions along a track [41]. Speed estimation was a dedicated task in the 2018 NVIDIA AI City Challenge [42], where participants combined deep learning-based detection with semi-automatic camera calibration techniques. Final vehicle speeds were derived using statistical measures such as mean, median, or percentile values of inter-frame displacement distances.

To further enhance speed estimation accuracy, several studies estimate 3D bounding boxes of vehicles by exploiting known vanishing point geometry and vehicle segmentation masks [16, 17]. In [3, 43], a rectified scene is first generated using detected vanishing points, after which a modified 2D object detector predicts an additional parameter alongside the bounding box to reconstruct the corresponding 3D bounding box. Another approach directly regresses 3D bounding box parameters based on centroid and vertex prediction using specialized neural networks [44]. Some researchers estimate speed by detecting license plates instead of entire vehicles [7, 18], leveraging induction loop data for validation.

The NVIDIA AI City Challenge 2018 [42] dataset consists of 27 high-definition, one-minute-long videos created specifically for evaluating vehicle speed estimation. However, ground truth speed annotations were not made publicly available, and participants were required to use a remote evaluation server.

## 3.Computationally Efficient Vehicle Speed Estimation

This section describes the proposed efficient vehicle speed estimation framework, which enhances the baseline approach by significantly reducing computational complexity. The original framework relied on the RetinaNet [5] object detection architecture, whereas we replace it with the more efficient YOLOv6 [4] model to achieve real-time performance on resource-constrained devices. This modification makes monocular vehicle speed estimation feasible in low-compute environments without compromising accuracy.

To ensure clarity and completeness, we first briefly summarize the baseline method introduced in [3], followed by a description of our improvements that lead to enhanced computational efficiency.

Overview of the Speed Estimation Pipeline

The complete pipeline includes the following major steps: Camera calibration Image rectification Vehicle detection 3D bounding box reconstruction Vehicle tracking Speed estimation Initially, the traffic camera is calibrated using the method described in [17], which identifies relevant vanishing points and computes scene scale to enable metric distance measurements on the road surface. Based on these vanishing points, a perspective transformation is applied so that two of the dominant scene directions align with the image axes. This rectification simplifies the representation of a vehicle's 3D bounding box, allowing it to be parameterized as a 2D bounding box plus one additional dimension-related parameter

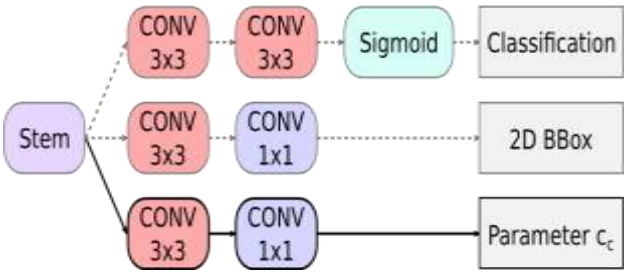
. The detection phase uses a modified object detector to produce these enhanced bounding boxes. The 3D bounding box is subsequently reconstructed in the original image space using the known vanishing point geometry. Vehicles are then tracked over time using a simple IoU-based tracker [38], and their speeds are computed from spatial displacement and calibration data. Our key improvement lies in replacing the original RetinaNet-based detection module with the more efficient YOLOv6 architecture, leading to significantly better computational performance compared to the baseline pipeline.

### 3.1 Baseline Method

The baseline framework presented in [3] estimates vehicle speeds by detecting 3D bounding boxes instead of traditional 2D bounding boxes. This approach provides a stable tracking reference point located at the center of the bottom frontal edge of the vehicle, ensuring accurate speed measurement even when camera angles vary.

The process begins with scene rectification based on prior camera calibration using vanishing point geometry [17]. A modified 2D object detector then predicts a standard bounding box along with an additional parameter required to reconstruct the corresponding 3D bounding box. These detected vehicles are tracked using a simple IoU-based tracking method [38]. For each vehicle track, speed is calculated using the detected spatial positions and calibrated camera geometry.

By introducing YOLOv6 in place of RetinaNet, our enhanced pipeline achieves superior computational efficiency while preserving the robustness and accuracy of the original method.

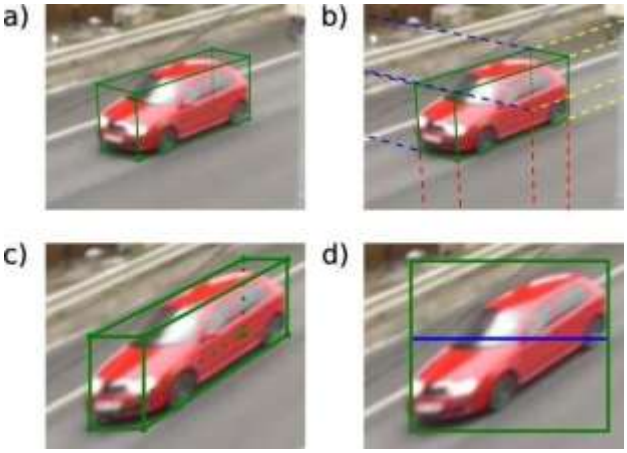



### 3.1 Improved base detector

To increase the computational efficiency of the baseline approach described in [3], we replace the RetinaNet [5] detector with YOLOv6 v3.0 [4]. YOLOv6 is an anchor-free object detection framework that follows a point-based detection strategy [46]. Instead of using predefined anchors, the network outputs a four-dimensional prediction consisting of distance distributions from the bounding box center to its edges, along with object classification and objectness confidence. These predictions are subsequently transformed into final bounding boxes through non-maximum suppression (NMS).

A major factor contributing to YOLOv6's fast inference performance is the RepVGG backbone architecture [47]. RepVGG uses structural reparameterization to streamline inference. During training, the network adopts a multi-branch structure inspired by ResNet [48], incorporating identity connections and 1×1 convolution branches. Once training is complete, these branches are algebraically merged into a single 3×3 convolution layer, integrating all parameters and batch normalization terms [49]. As a result, the final inference-stage model consists exclusively of 3×3 convolution layers followed by ReLU activations, making it highly optimized for GPU execution.

YOLOv6 is available in four different configurations: Nano, Small, Medium, and Large, enabling flexibility in balancing performance and computational cost. Although more recent models such as YOLOv7, YOLOv8, and YOLO-NAS provide marginal accuracy improvements, they introduce increased architectural and training complexity, which may not be suitable for low-compute edge environments.




The original YOLOv6 architecture outputs standard object class probabilities and 2D bounding box coordinates. To integrate the additional parameter $c$ introduced in [3] (see Figure 2), which is required for 3D bounding box reconstruction, we extend the prediction head by adding a custom output branch.

This added structure enables the model to simultaneously predict 2D bounding boxes and the extra geometric parameter required for 3D reconstruction. The modified prediction head is illustrated in Figure 3, where the original components are shown with dashed lines and the newly added layers with solid lines.

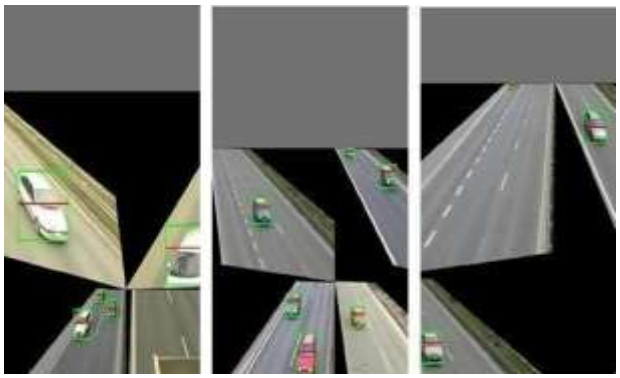


***Figure3:***

Illustration of the modified YOLOv6 prediction head. The original processing modules are depicted using dashed lines. To estimate the additional parameter $c$, a 3×3 convolutional block followed by a 1×1 convolutional layer is introduced, shown with solid lines.

***Figure2:***

When vehicles move along a straight roadway, their 3D bounding boxes (a) align with three dominant vanishing points (b). By identifying these points through automatic camera calibration (e.g., [17]), the scene can be rectified (c). In the rectified space, estimating a 3D bounding box is simplified to detecting a 2D bounding box with an added parameter $c$ (d), which specifies the location of the top front edge (blue) of the 3D box within the 2D projection (green). Image adapted from the original source.

***Figure4:***

Example showcasing mosaic data augmentation along with the corresponding annotated bounding boxes.

### 3.2Training data

3.3 Training Data Two datasets were used for training the proposed model: Noncapped [6] and BoxCars116k [52]. The Noncapped dataset consists of 21 traffic videos captured across seven distinct recording sessions. For

training and validation purposes, only the first four sessions were utilized.

To generate 3D bounding box annotations, we followed the methodology described in [3], which integrates vehicle segmentation masks obtained using [53] with the camera calibration data provided by the dataset. This process produces rectified images along with annotations represented as 2D bounding boxes enhanced with an additional parameter c (see Figure 2), required for reconstructing 3D bounding boxes.

Additionally, the BoxCars116k dataset was employed, which comprises images of individual vehicles along with corresponding camera calibration information and predefined 3D bounding boxes. These annotations were also converted into rectified images and bounding box representations compatible with our training pipeline.

### 3.3 Training Procedure

Instead of applying the preprocessing strategy suggested in [3], we adopted mosaic data augmentation during training. This technique involves randomly selecting four images from the training set and combining them into a single composite image arranged in a 2×2 grid. This merged image contains objects from all four original images, enhancing diversity and robustness. Furthermore, the images were randomly flipped and scaled, and their associated annotations (bounding boxes and parameter c) were adjusted accordingly. Examples of such augmented samples are shown in Figure 4.

All four versions of the YOLOv6 architecture (Nano, Small, Medium, and Large) were trained. Assignment of ground truth 2D bounding boxes to predicted outputs was performed using the TOOD strategy [54]. The regression of the additional parameter c was integrated into the total loss function using Mean Squared Error (MSE), defined as:

$$Lc = \frac{1}{} \sum (c_{c,i}^{p} - c_{c,i}^{g})^2 N$$

2 Each model was trained for 30 epochs, after which further improvement in validation loss was no longer observed. The first three epochs were used for warm-up. Optimization was carried out using Stochastic Gradient Descent (SGD) with a momentum of 0.937, a learning rate of 0.01, and a cosine annealing learning rate schedule.

After training, the model snapshot with the best validation performance was selected. Since the YOLOv6-Large model demonstrated superior validation results, we applied knowledge distillation to transfer its learned representations to the Nano and Small variants [26]. These distilled models were also trained for 30 epochs following the same optimization settings.

The validation results of all trained models are summarized in Table 1. It is worth noting that distillation led to a slight improvement in mean Average Precision (mAP) for the Nano model, whereas the distilled version of the Small model exhibited reduced performance compared to its non-distilled counterpart.

### 3.4 Post-training quantization

For model compression, we selected Post-Training Quantization (PTQ) instead of Quantization-Aware Training (QAT) due to its lower computational overhead and simpler implementation, making it more suitable for real-time vehicle speed estimation applications. This approach follows the workflow recommended by the YOLOv6 developers, where the trained floating-point ONNX model is converted into a TensorRT v8 engine using the default calibration procedure [26].

During calibration, a representative subset of images is passed through the network to gather activation statistics. These statistics are used to compute optimal scaling factors that map the FP32 numerical range into the reduced INT8 precision range. This process significantly reduces memory usage and inference latency while preserving detection accuracy.

For calibration, we utilized the training data described in Section 3.3, using a batch size of 32 across 32 calibration batches, resulting in a total of 1024 images used for calibration.

## 4.Results

This section presents the evaluation of the trained models in terms of vehicle speed estimation accuracy and computational performance across different configurations. Speed Measurement Evaluation

Vehicle speed estimation performance was assessed using the official evaluation tool provided by the creators of the BrnoCompSpeed dataset [6]. The accuracy results for Split C are summarized in Table 2. Among all tested configurations, the Nano and Small YOLOv6 models with an input resolution of 640 × 360 achieved the most balanced performance in both full precision and quantized modes.

For clearer comparison, the speed estimation errors of the best-performing models and the previous state-of- the-art approaches [3, 17] are illustrated using box plots in Figure 5.

Notable differences were observed in recall across the evaluated methods, which can introduce bias into speed accuracy comparisons since only correctly matched vehicle tracks are used for evaluation. Higher recall results in a larger and more diverse set of vehicle tracks, potentially increasing complexity. To address this, an additional experiment was conducted where all models were evaluated using the same subset of vehicle tracks that were successfully detected by every method. Results presented in Table 3 confirm that the quantized Nano model delivers superior speed estimation accuracy.

Interestingly, larger models did not outperform their smaller counterparts, despite achieving higher mAP and mAR values during validation (Table 1). This may be attributed to better generalization capabilities of smaller models or the fact that speed estimation relies on aggregated vehicle position data over multiple frames, making per-frame bounding box precision less critical. Furthermore, increasing the input resolution contributed to improved accuracy only up to 640 × 360, suggesting diminishing returns beyond this point.

In addition to Noncapped, our method was evaluated on the dataset introduced in [7] using the same evaluation metrics. As in [3], a separate model was trained for this dataset. The Small YOLOv6 model with a 640 × 360 input size and FP32 precision achieved a recall of 98.63%, with 93.95% of estimated speeds falling within the acceptable error range of −3 to +2 km/h. These results are comparable to those reported which achieved 98.88% recall and 92.71% accuracy within the same error bounds.

**Computational Efficiency**

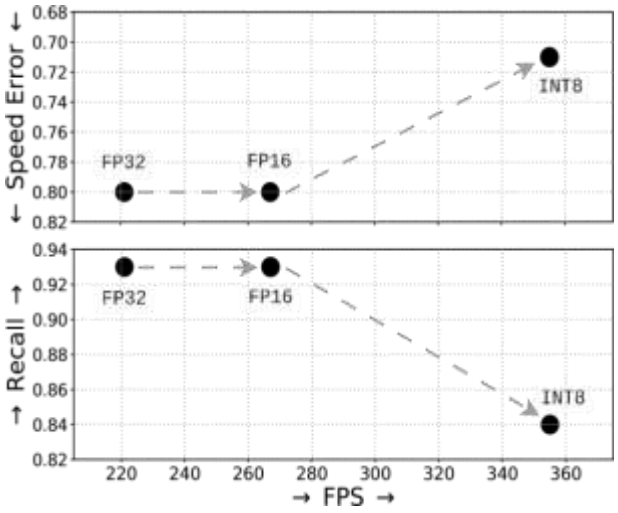
For real-world deployment, computational efficiency is a critical factor in vision-based vehicle speed estimation systems. Therefore, we conducted a comprehensive evaluation across multiple hardware configurations, including edge devices, consumer-level PCs, and high-performance servers. The GPUs used in the experiments are listed in Table 4.

When assessing the computational performance of the proposed pipeline, traffic density must also be taken into account, as higher traffic volumes increase the number of vehicles processed per frame, which in turn directly affects processing time and system throughput.

**5.Discussion**

This study introduces a computationally optimized framework for vehicle speed estimation by enhancing the baseline approach presented in [3]. The primary improvement lies in replacing the original object detector with a modified YOLOv6 architecture [4], resulting in a significant reduction in computational overhead while maintaining or improving estimation accuracy. Several variants of the proposed method were developed using different model sizes, and their real- world performance was evaluated across a broad range of hardware platforms, from edge devices to high-end desktop systems. The results demonstrate that the adoption of a more efficient detection architecture substantially reduces processing requirements while delivering comparable or superior vehicle speed measurement accuracy, along with improved detection precision and recall.

The notably lower computational demands make the proposed system suitable for deployment in resource-constrained environments, particularly on edge devices where real-time processing was previously impractical. This improvement broadens the applicability of vision-based vehicle speed estimation in modern intelligent transportation systems..




**References**

[1] J. Chen, Q. Wang, H. H. Cheng, W. Peng, and W. Xu, "Title not specified," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 11, p. 19954, 2022.

[2] X. Zhou, R. Ke, H. Yang, and C. Liu, "Title not specified," Applied Sciences, vol. 11, no. 20, p. 9680, 2021.

[3] V. Kocur and M. Ftáčnik, "Title not specified," Machine Vision and Applications, vol. 31, no. 7, p. 1, 2020.

[4] C. Li et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," arXiv preprint arXiv:2301.05586, 2023.

[5] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in Proc. IEEE Int. Conf. on Computer Vision (ICCV), 2017, pp. 2980–2988.

[6] J. Sochor et al., "On the Accuracy of Speed Measurement Using Visual Traffic Surveillance," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 5, pp. 1633–
1643, 2018.

[7] D. C. Luvizon, B. T. Nassu, and R. Minetto, "Vehicle Speed Estimation Using License Plate Detection," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 6, pp. 1393–1403, 2017.

[8] X. C. He and N. H. Yung, "Curvature Scale Space Corner Detection in Camera Calibration," in Proc. IEEE Workshop on Applications of Computer Vision, 2007, pp. 12–12.

[9] D. C. Luvizon et al., "Multiple-camera vehicle speed estimation," in Proc. IEEE ICASSP, 2014, pp. 6563–6567.

[10] C. Maduro et al., "Estimation of camera parameters for traffic monitoring," in Proc. IEEE ICIP, 2008, pp. 777–780.

[11] X. You and Y. Zheng, "Automatic camera calibration method," Neurocomputing, vol. 204, pp. 222–231, 2016.