Real-Time Deep Learning Framework for Accurate Speed Estimation of Surrounding Vehicles in Autonomous Driving

Iván García-Aguilar 1,2, Jorge García-González 1,2, Enrique Domínguez 1,2, Ezequiel López-Rubio 1,2 and Rafael M. Luque-Baena 1,2,*

1 Institute of Software Technologies and Engineering (ITIS), University of Málaga, C/Arquitecto Francisco Peñalosa, 18, 29010 Málaga, Spain; ivangarcia@uma.es (I.G.-A.); jorgegarcia@uma.es (J.G.-G.); enriqued@uma.es (E.D.); elr@uma.es (E.L.-R.)
2 Biomedical Research Institute of Málaga (IBIMA), C/Doctor Miguel Díaz Recio, 28, 29010 Málaga, Spain
* Correspondence: rmluque@uma.es

Abstract: Accurate speed estimation of surrounding vehicles is of paramount importance for autonomous driving to prevent potential hazards. This paper emphasizes the critical role of precise speed estimation and presents a novel real-time framework based on deep learning to achieve this from images captured by an onboard camera. The system detects and tracks vehicles using convolutional neural networks and analyzes their trajectories with a tracking algorithm. Vehicle speeds are then accurately estimated using a regression model based on random sample consensus. A synthetic dataset using the CARLA simulator has been generated to validate the presented methodology. The system can simultaneously estimate the speed of multiple vehicles and can be easily integrated into onboard computer systems, providing a cost-effective solution for real-time speed estimation. This technology holds significant potential for enhancing vehicle safety systems, driver assistance, and autonomous driving.

Keywords: speed estimation; deep learning; convolutional neural networks; regression

1. Introduction

The impact of the actual Advanced Driver Assistance Systems (ADAS) is profound in today’s automotive landscape. ADAS contributes to a reduction in road fatalities by minimizing human errors [1,2]. However, pursuing even higher levels of autonomy (specifically, self-driving vehicles rather than mere driver support features) remains an ambitious challenge.

We must delve into the intricacies of subsystems within the perception and decision-making engines to achieve this goal. These subsystems operate under adverse conditions, including lightning, severe weather, and complex traffic scenarios. The Society of Automotive Engineers (SAE) has meticulously defined six levels of driving automation. At the pinnacle of this hierarchy lies Level 5, denoting a vehicle capable of navigating all situations without human intervention [3].

To ascend to these lofty levels of automation, robust perception algorithms and anomaly detection methods are essential. Additionally, assimilating vast amounts of data (coupled with continuous learning) is crucial. The journey toward Level 5 automation remains exciting and challenging as we strive for safer, more autonomous transportation.

Tracking surrounding vehicles is pivotal as a critical subsystem within perception tasks. Its significance extends across various features, including lane detection, active brake systems, and identifying hazardous maneuvers [4,5]. The field of target tracking has undergone extensive study, particularly in the context of surveillance and defense-related applications [6–9]. Research in this area delves into complex challenges related to multi-target tracking, extent estimation, and track-before-detect methods [8,10,11]. In these scenarios, RADAR emerges as the sensor of choice.
Modern automobiles come equipped with two types of automotive RADAR sensors: short-range (SRR) and long-range (LRR). These sensors operate in the 24 GHz and 76–81 GHz frequency bands, respectively. While RADAR sensors remain unaffected by adverse weather conditions such as lightning, fog, and rain, their performance can be significantly compromised. The prevalence of metallic surfaces in automotive environments leads to cluttered signals and low signal-to-noise ratios (SNR).

As an appealing alternative to RADAR, LiDAR sensing offers several advantages. LiDAR is less susceptible to clutter noise and provides substantially higher resolution. However, the current production costs of LiDARs render them prohibitively expensive for most vehicles. Additionally, integrating LiDAR data necessitates specialized graphical processing [12].

In response to these challenges, windshield-installed cameras are gaining popularity. These cameras capture images that are subsequently processed to identify lane markings, pedestrians, and speed limits on traffic signs [13]. Moreover, user experience design increasingly relies on vision-based stimuli rather than traditional RADAR information [14].

In the dynamic realm of automobile technology, the domain of target tracking within video sequences remains surprisingly underexplored. Despite the growing popularity of camera-based algorithms, the intricacies of video-based target tracking have not received the attention they deserve. This gap in research motivated our investigation into camera-based relative speed estimation.

Our innovative approach centers around a single windshield-mounted camera, which captures images of the surrounding environment. An object detector then meticulously processes these images, allowing us to extract precise vehicle bounding boxes. The centroids of these bounding boxes serve as key reference points for tracking the vehicles as they move through the scene. By analyzing changes in their size over time, we can infer the relative velocity of the tracked vehicles. Remarkably, this entire process occurs autonomously, without any human intervention.

Our contribution to the field is significant: we present a video-based relative speed estimation solution. What sets our approach apart is its adaptability. We allow flexibility in choosing the vehicle detection algorithm, ensuring compatibility with various scenarios and sensor setups. Additionally, our method exhibits resilience against unwanted outliers and inaccuracies thanks to our utilization of robust estimation techniques.

In summary, our work bridges a critical gap in the study of target tracking, unlocking new possibilities for safer and more efficient automotive systems.

The remainder of this paper is structured as follows. Section 2 details the state of the art and related works in the field of the presented methodology. Section 3 introduces the convolutional neural networks for detection, the regression models, and the proposed methodology outlined in this article. Section 4 describes the experiments and results, including the selected dataset, the metrics used for evaluation, and the quantitative and qualitative results. Section 5 discusses the results. Finally, Section 6 comprises the conclusions and future directions.

2. Related Works

The first step of this proposal is to detect vehicles within an image obtained by an onboard camera. Generic object detection is a long-standing problem that has been widely explored in computer vision. Classical object detection methods using manually designed features could be used, but models based on deep convolutional neural networks trained to infer features are faster and more accurate automatically. These methods can be divided into two main paradigms: two-stage detection models and one-stage detection models. Methods from the first set first propose region candidates where there could be an object and then apply an object classifier model to determine if there is any object and its class. Examples are CenterNet [15], R-CNN [16], Fast R-CNN [17], Faster R-CNN [18], and EfficientNet [19]. Although the accuracy of these methods is usually high, their two-step paradigm is computationally demanding and requires more time to perform the detection.
The reason for discarding them is the time-dependent nature of the proposal. Two-stage models include SSD (single-shot multi-box detector) and YOLO (you only look once). These models create candidate regions and their detected objects within a single process flow, avoiding unnecessary calculations. Usually, they are faster than two-stage models but with lower accuracy or more instability. However, YOLO’s latest proposals are not only fast but also reliable. This makes them the default candidate when it comes to real-time applications.

After detecting vehicles, our proposal must track them for a long time. The same vehicle in consecutive video frames must be identified as the same vehicle. This problem has been studied within computer vision for surveillance tasks for decades. Although deep learning solutions for this problem are viable, classical methods are also competitive ways to solve it. The SORT algorithm tracks multiple objects simultaneously and online using state estimation techniques and data association. DeepSORT is an extension that also uses appearance features obtained via deep learning. Other options like [20] use backtracking to refine anomalies. Combining DeepSORT, YOLO, and non-maximum-suppression is used by [21] to filter trajectories.

Speed estimation is the last step once vehicles are detected and properly tracked. Several ways of dealing with this problem have been proposed when working with static cameras. Homographic transformations based on manually selected reference points to correct the camera’s perspective are widely used to correctly measure distances properly and, thus, speeds [22]. Ref. [23] uses evolutionary algorithms to align points between planes and adapt to perspective changes. Neural networks are also used to measure speed. Ref. [24] proposes the creation of a perspective transformation based on a geometry of vanishing points. In that way, from 2D, bounding boxes can be extended to 3D to obtain speeds. When the camera is not static, the problem is more complex since most options applied to the static problem are not useful. This is why, when working with measures from a moving vehicle, more specific hardware like RADAR and LiDAR is usually required. Ref. [25] summarizes the general approach to vehicle speed estimation from static traffic or speed cameras. The general approach implies obtaining a relation between pixels and meters, usually obtained from camera intrinsic and extrinsic parameters in combination with terrain information. Ref. [26] is an example of this approach.

Another typical approach is to rely on specific hardware such as LiDAR to perform the speed estimation [27–29]. Even if these approximations are the most accurate and have been extensively studied in the literature, their major problem is that they rely on hardware systems useful only for measuring distances and with a high monetary cost.

Synthetic datasets have become increasingly prevalent in validating vehicle detection and speed estimation systems. The CARLA simulator, as discussed by Dosovitskiy et al. [30], has provided a versatile platform for generating realistic driving scenarios, enabling extensive testing and validation of autonomous driving algorithms. This approach has been adopted by several studies, such as Ros et al. [31] and Gaidon et al. [32], to augment training data and enhance model performance.

3. Models and Methods

In this section, the proposed methodology is denoted as RT-VE (real-time vehicle estimation) is described in detail. The provided method is shown in Figure 1. The methodology is designed to accurately estimate the speed of surrounding vehicles in real time, leveraging deep learning techniques. Initially, images captured by an onboard camera are used as input. Convolutional neural networks (CNNs) are employed to detect and track vehicles within these images. The detected trajectories are then analyzed using a robust tracking algorithm. Vehicle speeds are estimated using a regression model based on random sample consensus (RANSAC). A synthetic dataset has been generated to validate the presented methodology, ensuring the system’s reliability and accuracy. The framework is capable of simultaneously estimating the speed of multiple vehicles and can be seamlessly integrated into onboard computer systems, offering a cost-effective solution for real-time speed estimation. This technology has significant potential applications in vehicle safety systems,
driver assistance, and autonomous driving. Below, each of the components comprising the proposed methodology is detailed.

**Figure 1.** Flow diagram of the proposed approach.

### 3.1. Convolutional Neural Networks for Object Detection

The object detection step is the most computationally heavy of all the processes in the proposed methodology. To achieve real-time, the YOLO model was selected following other state-of-the-art proposals, such as [33]. Identifying vehicles whose speed must be estimated requires the application of an object detection model in real time. For this purpose, YOLOv7x (you only look once version 7X) [34] has been utilized. Nevertheless, the methodology presented is not exclusively dependent on this specific model, allowing for the potential use of other models. YOLOv7x represents a significant advancement within the YOLO model family, renowned for its exceptional accuracy and real-time processing capabilities. Starting from a pre-trained state with the COCO dataset [35], this model has demonstrated the ability to rapidly and efficiently identify and locate vehicles in high-traffic areas under a wide range of conditions.

The operation of the YOLOv7x model is predicated on dividing the input image into a grid. Subsequently, each cell within the grid simultaneously predicts a series of bounding boxes and the probability of the class to which each detected object belongs. The architecture of this model is meticulously designed for real-time processing, enabling high-speed detection of numerous objects in a single pass. This capability is crucial for scenarios that demand immediate and precise object detection, such as autonomous driving systems.

Integrating this model with a tracking algorithm makes it feasible to uniquely identify and locate vehicles in each frame of the video sequence under evaluation. This integration is essential for determining the position of vehicles over time, thereby establishing the foundation for the subsequent speed estimation phase. The speed estimation is based on analyzing the temporal changes in the bounding box that encloses each vehicle. As the bounding box’s dimensions and position evolve over consecutive frames, the tracking algorithm provides the necessary data to accurately calculate the vehicle’s speed.
3.2. Regression Models

The random sample consensus algorithm (RANSAC) [36] is one of the most robust methods for estimating model parameters from a dataset containing a significant proportion of outliers. Unlike several traditional techniques, which outliers can highly influence, RANSAC performs an iterative selection of randomly generated subsets, fitting the model. It then performs a consensus counting of the number of outliers, determined as those points that fit well within a specified margin of error. In this way, it is possible to identify the model that best represents most of the data, thus effectively minimizing the impact of outliers.

In scenarios based on driver support systems, the data captured by onboard cameras may contain several inaccuracies due to factors such as changing lighting conditions and occlusions, which can influence the determination of the exact location of vehicles, thus introducing some outliers into the dataset. This can significantly bias the results. By employing RANSAC, this estimation process becomes more robust in obtaining reliable estimates. Although RANSAC is suitable for this application, the use of other regression models is not mutually exclusive. The methodology is designed to be flexible, allowing the substitution of such a model for a more appropriate one according to the scope of the application.

3.3. Methodology

This section outlines our approach to identifying vehicles moving at dangerous speeds. Prior to deployment in real-world scenarios, the methodology underwent extensive testing on a computer platform. Synthetic video data were utilized to simulate diverse driving scenarios within the CARLA simulator, a tool designed for autonomous driving research. This simulated environment mimics urban settings with varying traffic dynamics. The processing was conducted under conditions similar to those expected in deployment, leveraging the computer’s higher computational capabilities to facilitate parallel testing and model refinement. This approach ensured that the detection and tracking algorithms were thoroughly validated and optimized before transitioning to field trials and actual deployment scenarios. Initially, we utilize a deep convolutional neural network to perform object detection on each video frame, producing a list of detected vehicles and their bounding boxes for each frame. Subsequently, we employ an object-tracking method to ascertain the trajectories of the vehicles. A trajectory is considered as a sequence of a vehicle’s positions across video frames, marked by the corresponding bounding boxes, acknowledging that some intermediate frames may lack detections if the vehicle was not identified in those frames.

Let \( t \) denote the current frame index within the acquired video sequence. Additionally, let \( \delta \) represent the angular diameter of an object, expressed in radians, i.e., the apparent diameter. Then we have the following:

\[
\delta = 2 \arctan \frac{d}{2D}
\]  

where \( D \) represents the distance from the camera to the object of interest, and \( d \) is the actual diameter of the object, both measured in meters. From (1), we have the following:

\[
\frac{d}{2 \tan \frac{\delta}{2}} = D
\]  

In practice, the approaching vehicle is visible for a brief period. Thus, it can be assumed that its speed, \( v \), remains constant relative to the camera during this interval:

\[
D = e_0 + vt
\]  

where \( e_0 \) represents the distance at time \( t = 0 \) to the other vehicle.
Given the assumption that the speed is constant, from (2) and (3), the following can be obtained:

\[
\frac{d}{2 \tan \frac{\delta}{2}} = e_0 + vt
\]

(4)

Additionally, let us assume that the distance from the camera to the vehicle is significantly larger than the vehicle’s size, i.e., \(D \gg d\). Under this condition, the small angle approximation \(a \approx \arctan a\) can be applied to (4). This results in the following:

\[
\frac{1}{\delta} = \frac{e_0 + vt}{d}
\]

(5)

It is essential to highlight that \(d\) and \(e_0\) are assumed constant for each vehicle. Therefore, from Equation (5), it can be obtained that the inverse of the apparent diameter \(1/\delta\) is linearly related to the time index \(t\). A practical approach to approximating the apparent diameter \(\delta\) of a vehicle is to take the square root of the number of pixels (area) in its bounding box.

Let \(y\) denote the collected samples for the approximation, i.e., the values of \(1/\delta\). Using this information, the slope of the line associated with Equation (5) can be computed via linear regression. This slope defines the speed \(v\) of the incoming vehicle, relative to the onboard camera. For each frame of the acquired video where the incoming vehicle is detected, a sample is collected for the linear regression method. It can be assumed without loss of generality that \(d = 1\) in Equation (5) for the sake of performing the linear regression. However, this implies that the relative speeds \(v\) are not calibrated.

After the linear regression is carried out, the computation of a calibration constant \(K\) is performed by comparison of the non-calibrated speeds \(v\) with the ground truth speeds. Once this procedure is completed, the uncalibrated speeds should be multiplied by \(K\) to yield the calibrated estimated speeds.

The random sample consensus (RANSAC) linear regression method has been utilized to estimate the speed \(v\). This method automatically identifies and excludes outliers \(y_i\) within the context of a standard linear regression. The RANSAC algorithm for linear regression includes the following steps:

1. Draw a random subset \(S\) of \(n\) data samples:

\[
S \subseteq \{(x_i, y_i)\}_{i=1}^{N}
\]

(6)

2. A linear model is fit to the following subset:

\[
y = \beta_0 + \beta_1 x
\]

(7)

where the coefficients \(\beta_0\) and \(\beta_1\) are found by minimization:

\[
\min_{\beta_0, \beta_1} \sum_{(x_i, y_i) \in S} (y_i - (\beta_0 + \beta_1 x_i))^2
\]

(8)

3. Inliers are identified, i.e., data samples from the complete set that are found to lie within a threshold \(\epsilon\) of the model:

\[
inliers = \{(x_i, y_i) \mid |y_i - (\beta_0 + \beta_1 x_i)| < \epsilon\}
\]

(9)

4. If the inlier count \(|\text{inliers}|\) exceeds a preset threshold \(T\), then the model must be fit again, this time employing all inliers:

\[
\min_{\beta_0, \beta_1} \sum_{(x_i, y_i) \in \text{inliers}} (y_i - (\beta_0 + \beta_1 x_i))^2
\]

(10)
5. Repeat the preceding steps 1–4 for a preset iteration count \( k \) or until convergence.

The final model is the best one that has been encountered through the execution of the loop:

\[
\hat{g} = \hat{\beta}_0 + \hat{\beta}_1 x
\]  

(11)

The final stage of our method involves applying a threshold to the calibrated estimated speeds. The purpose of this procedure is to identify and flag approaching vehicles with excessively high speeds as dangerous.

4. Experiments and Results

Below, the selected dataset is presented, along with the evaluation metrics established to verify the robustness of the proposed methodology. Finally, the results obtained are shown.

4.1. Dataset

CARLA [30], an open-source simulator for research in the autonomous driving environment, has been used to generate the required data for this study. It provides a realistic and flexible environment that allows the generation of various driving scenarios. Thus, it offers many advantages in simulating complex urban environments, such as applying multiple weather conditions or creating specific traffic scenarios. The simulator is an ideal tool for generating large and customizable synthetic data sets based on this flexibility.

A 20-min driving sequence has been generated to analyze the results obtained after applying the proposed methodology, which contains a series of vehicles circulating on the road performing a series of maneuvers. The generated driving sequence represents a typical urban environment with a mix of straight roads, intersections, and roundabouts to reflect common driving conditions. The simulation was conducted under clear weather conditions to ensure consistent data quality. The vehicles within the scenario performed a variety of maneuvers, including acceleration, deceleration, and lane changes, to capture a broad spectrum of driving behaviors. Each vehicle’s speed and position data were recorded at a high frequency, providing detailed ground truth information for every frame. This detailed data collection allows for precise validation of the speed estimation model. A single, well-defined scenario ensures controlled conditions, minimizing external variables that could affect the results, and allows for a focused assessment of the model’s performance.

During this simulation, the real data of the speed of each vehicle expressed in meters per second has been recorded for each frame. This information makes it possible to accurately validate the model by estimating the established speeds. The use of synthetic data generated by Carla is particularly advantageous in this context because, at the moment, there are no publicly available datasets that provide complete real scenarios with the exact actual speeds of each vehicle and of the vehicle capturing the images at each instant. Another advantage of using this simulator is that it allows large volumes of information generation without the ethical concerns associated with real-world data collection. An example of one of the frames that make up the sequence is shown in Figure 2.
4.2. Metrics

A carefully selected set of metrics was employed to evaluate the performance of the velocity estimation method. These metrics were chosen to provide a comprehensive assessment of actual and estimated velocities and the accuracy and consistency of the model predictions.

The mean speed metric serves as a baseline measure of the actual and estimated speeds, helping contextualize the model’s performance. Alongside this, the median speed offers a robust measure of central tendency less sensitive to outliers than the mean, giving a more accurate representation of typical speeds. The standard deviation of both the real and estimated speeds is used to indicate variability, allowing us to see the range and distribution of the speeds. This helps understand the consistency and spread of the speed values in the dataset.

Regarding error metrics, the mean absolute error provides a straightforward measure of prediction accuracy by indicating the average error magnitude in the model’s speed estimations. Complementing this, the mean squared error emphasizes more significant errors due to the squaring process, offering a more sensitive measure of prediction accuracy. Similarly, the median absolute error provides a robust measure of prediction accuracy less affected by outliers, indicating the typical prediction error. While similar to the mean squared error, the median squared error is more robust to outliers, providing a balanced measure of more significant errors.

Additionally, the standard deviation of error indicates the variability in the prediction errors, which helps understand the consistency of the model’s performance. Finally, the coefficient of determination measures how well the estimated speeds approximate the real speeds, with a value closer to 1 indicating a better fit for the model.

4.3. Results

This section presents a detailed analysis of the speed estimation for several vehicles. Tests have been conducted using synthetic data on a high-performance computer to evaluate the accuracy and performance of the detection model. The experiments were performed on a system with an NVIDIA (Santa Clara, CA, USA) GeForce RTX 3080 GPU and 64 GB of RAM, ensuring robust computational capabilities. Despite these tests being conducted in a controlled, high-resource environment, the detection model has been specifically adapted for real-time execution on low-cost devices embedded directly within the vehicle. This adaptation is crucial for practical deployment scenarios where computational resources are limited. The decision to utilize YOLO as the detection algorithm stems from its efficiency.
and effectiveness in real-time object detection, making it well-suited for implementation on embedded systems.

The experiments utilized an object detection model to conduct the numerical studies. The YOLO (you only look once) algorithm was employed explicitly for real-time object detection tasks. Several assumptions and boundary conditions were established for the experiments. Firstly, it was assumed that the synthetic data used for training and testing accurately represented real-world scenarios, including variations in vehicle speed, lighting conditions, and environmental factors. The boundary conditions included setting vehicles’ initial positions and speeds within a predefined range to ensure consistency across all tests. Moreover, it was assumed that the detection model would operate under controlled conditions, meaning that extreme weather conditions or highly irregular road surfaces were not part of the testing scenarios. The model was also tested assuming that the input data had minimal noise, although some degree of noise and anomalies were considered to evaluate the model’s robustness. Finally, the model’s performance was assessed, assuming it would be deployed on low-cost devices with limited computational resources, which guided the choice of the YOLO algorithm for its efficiency and effectiveness in real-time processing.

The analysis compares the ground truth (GT) speeds with the estimated speeds, and various statistical measures are computed, including the absolute error, mean squared error (MSE), median absolute error, median squared error, and coefficient of determination ($R^2$). Table 1 provides a summary of the statistical measures for each vehicle.

Table 1. Comparison of real and estimated vehicle speeds (m/s) with error metrics for several vehicles.

<table>
<thead>
<tr>
<th>Car</th>
<th>Real Speed</th>
<th>Estimated Speed</th>
<th>Absolute Error</th>
<th>Mean Squared Error</th>
<th>Median Absolute Error</th>
<th>Median Squared Error</th>
<th>Coefficient of Determination</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.09 ± 5.86</td>
<td>11.05 ± 7.21</td>
<td>3.29</td>
<td>31.59</td>
<td>2.08</td>
<td>4.34</td>
<td>0.08</td>
</tr>
<tr>
<td>22</td>
<td>11.68 ± 5.21</td>
<td>11.73 ± 6.64</td>
<td>2.64</td>
<td>27.49</td>
<td>1.12</td>
<td>1.24</td>
<td>−0.01</td>
</tr>
<tr>
<td>50</td>
<td>10.78 ± 4.79</td>
<td>11.12 ± 6.47</td>
<td>3.11</td>
<td>27.04</td>
<td>1.63</td>
<td>2.65</td>
<td>−0.18</td>
</tr>
<tr>
<td>53</td>
<td>6.75 ± 3.69</td>
<td>6.39 ± 4.19</td>
<td>2.81</td>
<td>21.46</td>
<td>1.35</td>
<td>1.82</td>
<td>−0.58</td>
</tr>
<tr>
<td>56</td>
<td>2.09 ± 2.06</td>
<td>2.90 ± 2.12</td>
<td>2.26</td>
<td>11.85</td>
<td>1.19</td>
<td>1.42</td>
<td>−1.80</td>
</tr>
<tr>
<td>59</td>
<td>7.86 ± 4.35</td>
<td>8.53 ± 6.51</td>
<td>3.18</td>
<td>28.21</td>
<td>1.71</td>
<td>2.93</td>
<td>−0.49</td>
</tr>
<tr>
<td>60</td>
<td>6.90 ± 6.15</td>
<td>7.40 ± 6.88</td>
<td>2.60</td>
<td>23.38</td>
<td>0.55</td>
<td>0.30</td>
<td>0.38</td>
</tr>
<tr>
<td>82</td>
<td>5.94 ± 3.31</td>
<td>6.65 ± 3.82</td>
<td>2.43</td>
<td>26.56</td>
<td>0.96</td>
<td>0.92</td>
<td>−1.42</td>
</tr>
<tr>
<td>122</td>
<td>11.08 ± 5.56</td>
<td>10.97 ± 6.81</td>
<td>2.32</td>
<td>17.34</td>
<td>1.22</td>
<td>1.50</td>
<td>0.44</td>
</tr>
<tr>
<td>125</td>
<td>8.53 ± 5.39</td>
<td>9.55 ± 6.42</td>
<td>3.21</td>
<td>30.34</td>
<td>1.80</td>
<td>3.23</td>
<td>−0.04</td>
</tr>
<tr>
<td>143</td>
<td>11.34 ± 6.56</td>
<td>12.02 ± 7.07</td>
<td>2.70</td>
<td>20.94</td>
<td>1.42</td>
<td>2.01</td>
<td>0.51</td>
</tr>
<tr>
<td>148</td>
<td>5.28 ± 4.31</td>
<td>4.81 ± 4.68</td>
<td>1.73</td>
<td>9.45</td>
<td>0.78</td>
<td>0.61</td>
<td>0.49</td>
</tr>
<tr>
<td>157</td>
<td>9.60 ± 4.22</td>
<td>9.47 ± 4.88</td>
<td>2.05</td>
<td>12.74</td>
<td>1.11</td>
<td>1.23</td>
<td>0.29</td>
</tr>
<tr>
<td>166</td>
<td>6.63 ± 2.73</td>
<td>7.01 ± 4.75</td>
<td>2.29</td>
<td>25.35</td>
<td>0.74</td>
<td>0.54</td>
<td>−2.40</td>
</tr>
<tr>
<td>176</td>
<td>9.67 ± 4.24</td>
<td>9.78 ± 6.27</td>
<td>3.11</td>
<td>27.76</td>
<td>1.94</td>
<td>3.75</td>
<td>−0.54</td>
</tr>
</tbody>
</table>

The estimated speeds are generally close to the real speeds, with some deviations. For example, vehicle 10 has an average real speed of 10.09 ± 5.86 and an estimated speed of 11.05 ± 7.21. This indicates that the estimation method performs reasonably well in general. According to the absolute and mean squared error, it varies among the vehicles. For instance, vehicle 56 has a relatively high Absolute Error of 2.26 and an MSE of 11.85, suggesting significant deviations in speed estimation for this vehicle.

The median errors provide additional insight into the performance of the estimation method. Vehicle 60 shows a low median absolute error of 0.55 and a median squared error of 0.30, indicating accurate estimation for the majority of the data points.

Finally, the coefficient of determination varies significantly, with several vehicles showing positive $R^2$ values, indicating a good fit, while others have negative values, suggesting poor performance in those cases. For instance, vehicle 166 has an $R^2$ of −2.40, indicating a poor fit.
Figures 3–6 provide a visual representation of the speeds for selected vehicles:

**Figure 3.** Comparison between actual and estimated speeds for the vehicle with ID 122.

The speeds for vehicle 122 are shown in Figure 3. There is a good alignment between the GT and estimated speeds, although some discrepancies are observed, particularly in the middle section of the frame range.

**Figure 4.** Comparison between actual and estimated speeds for the vehicle with ID 148.

Figure 4 depicts the speeds for vehicle 148. The GT and estimated speeds show a high level of correspondence throughout the frames, with minor deviations. The plot suggests that the estimation method performs well for this vehicle.
Figure 5. Comparison between actual and estimated speeds for the vehicle with ID 157.

Figure 5 illustrates the speeds for vehicle 157. The GT and estimated speeds are closely aligned for the majority of the frames. However, there are spikes in the estimated speed that do not correspond to the GT speed, indicating some outliers or noise in the estimation process.

Figure 6. Comparison between actual and estimated speeds for the vehicle with ID 166.

In Figure 6, the GT speed (red) and the estimated speed (green) for vehicle 166 are plotted against the frame number. The plot shows periods of close alignment between the real and estimated speeds, with occasional significant deviations, particularly at higher speeds.

In Figure 7, an example is presented in which the speed of a vehicle is estimated. This illustration highlights the methodology employed to calculate the velocity, providing a clear demonstration of the estimation process in practical application.
5. Discussion

Analyzing the speed estimation for several vehicles reveals several key insights and areas for improvement. This section discusses the implications of the statistical results and figures, as well as potential factors influencing the performance of the estimation method.

One of the most prominent observations from the results is the variability in the estimation performance across different vehicles. While the mean estimated speeds generally align closely with the real speeds, the errors and the coefficient of determination ($R^2$), values exhibit considerable variation. Vehicles such as 122 and 143 show relatively high $R^2$ values (0.44 and 0.51, respectively), indicating a good fit between the real and estimated speeds. Conversely, vehicles like 166 and 56 have significantly negative $R^2$ values ($-2.40$ and $-1.80$, respectively), suggesting poor performance.

The variability in performance can be attributed to several factors:

- **Vehicle speed dynamics:** Vehicles with more dynamic and variable speeds may pose a greater challenge for the estimation method, leading to higher errors. This is evident in vehicles like 56, with high absolute and mean squared errors.

- **Sensor and environmental factors:** The quality of sensor data and environmental conditions such as lighting, weather, and road conditions can impact speed estimation accuracy. Variations in these factors across different vehicles and frames may contribute to discrepancies in the results.

- **Algorithm limitations:** The underlying assumptions and limitations of the estimation algorithm itself could result in varying accuracy. For example, if the algorithm is more sensitive to specific speed ranges or vehicle types, this could explain the observed performance differences.

According to the error analysis metrics, absolute, mean squared error (MSE), median absolute, and median squared error provide a comprehensive picture of the estimation accuracy. Vehicles with lower errors, such as 148 and 60, indicate that the estimation method is highly reliable for those cases. In contrast, vehicles like 166 and 56, with higher errors, highlight potential outliers or specific scenarios where the method struggles.

- **Absolute and median errors:** Lower median errors in vehicles such as 60 (Median absolute error of 0.55) suggest that the estimation method performs consistently well for most frames. However, higher absolute errors indicate that there are instances with significant deviations.

- **Squared errors:** The MSE and median squared error metrics emphasize the impact of larger deviations. For instance, vehicle 56 has an MSE of 11.85, reflecting the influence of a few large errors on the overall performance.

Figures provide visual evidence of the alignment and discrepancies between the ground truth and estimated speeds, serving as a diagnostic tool to identify specific frames or periods where the estimation method succeeds or fails. For instance, Figure 1 (Vehicle 166) reveals both good alignment and significant deviations, particularly at higher speeds, suggesting challenges with fluctuating speeds. Figure 2 (Vehicle 157) indicates generally
good performance with occasional outliers. Figure 3 (Vehicle 148) demonstrates high effectiveness with minor deviations, while Figure 4 (Vehicle 122) shows good overall alignment but highlights specific segments with potential challenges. These visual insights are crucial for understanding and improving the estimation method’s performance.

The discrepancies between the real and predicted values can be attributed to several factors. One primary factor is vehicle speed’s dynamic and variable nature, which can pose a significant challenge for the estimation model. Vehicles that exhibit abrupt or frequent speed changes, as observed in the cases of vehicles 56 and 166, tend to generate higher errors due to the model’s difficulty in quickly adapting to these changes. Additionally, environmental conditions play a crucial role. Factors such as lighting, weather conditions, and road surface quality can impact the accuracy of the input data, leading to discrepancies in the estimated speeds. The algorithm’s capacity to handle noise or anomalies in the data is also limited; spikes or deviations in the estimated speeds, as seen in vehicle 157, suggest the presence of noise or anomalous data that the model fails to filter adequately. Finally, inherent limitations of the estimation algorithm, especially if it is not optimized for all speed ranges or vehicle types, can contribute to variability in performance.

Based on the analysis, several recommendations can be made to enhance the speed estimation method:

- **Algorithm refinement**: Further refining the estimation algorithm to better accommodate dynamic speed changes and reduce sensitivity to outliers could improve accuracy. Incorporating machine learning techniques to adapt to different vehicle dynamics might be beneficial.
- **Error handling and correction**: Implementing error correction mechanisms, such as filtering techniques to smooth out spikes or anomalies in the estimated speeds, could enhance overall performance.
- **Contextual adaptation**: Developing context-aware algorithms that adjust their parameters based on real-time conditions (e.g., road type, traffic density) could lead to more accurate and reliable speed estimation.

### 6. Conclusions and Future Work

In this work, based on vehicles detected from a single onboard camera, the speeds of those other vehicles are estimated using a linear regression model. The proposal requires no other input or hardware to measure distance or relative speed, so we consider it a promising research line to obtain an alternative to other highly costly hardware such as LIDAR.

Experiments have been carried out using data obtained through the CARLA simulator. This approach was selected due to the difficulty of obtaining actual data with ego and relative velocity for all vehicles. Experiments show that the proposal works within a reasonable error margin, given the limited information used to obtain the estimation.

As stated, the proposal is based on detections made using an object detection model. Improvements in this key element would imply greater stability in tracking the vehicles and their speeds. Also, including error handling during the tracking phase could lead to fewer anomalies in the speed estimation. Other techniques to detect and smooth those anomalies could be explored. However, anomaly detection needs to be online to work in real-time as a drive assistant system, and very few samples after the anomaly event are required not to slow the detection of genuine speed spikes.

**Funding:** This work is partially supported by the Ministry of Science and Innovation of Spain, grant number PID2022-136764OA-I00, project name Automated Detection of Non-Lesional Focal Epilepsy by Probabilistic Diffusion Deep Neural Models. It includes funds from the European Regional Development Fund (ERDF). It is also partially supported by the University of Málaga (Spain) under grants B1-2022_14, project name Detección de trayectorias anómalas de vehículos en cámaras de tráfico; and, by the Fundación Unicaja under project PUNI-003_2023, project name Intelligent System to Help the Clinical Diagnosis of Non-Obstructive Coranary Artery Disease in Coronary Angiography.

**Data Availability Statement:** The data supporting this study are publicly available in the GitHub repository at https://github.com/IvanGarcia7/SpeedCARLADataset (accessed on 15 July 2024).

**Acknowledgments:** The authors thankfully acknowledge the computer resources, technical expertise, and assistance provided by the SCBI (Supercomputing and Bioinformatics) Center of the University of Málaga. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of an RTX A6000 GPU with 48 Gb. The authors also thankfully acknowledge the grant of the Universidad de Málaga and the Instituto de Investigación Biomédica de Málaga y Plataforma en Nanomedicina-IBIMA Plataforma BIONAND.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.