Tracking and Estimating the Speed of Vehicles Using an Enhanced R-CNN and Deep SORT Architecture

Marwa. A. Marzouk [1,*] and Mohamed Elkholy [2,*]

[1] Faculty of Computers and Artificial intelligence, Matrouh University, Egypt
[2] Faculty of Computer Science & Engineering, Alamein International University, Egypt
Email: Marwa.Abdel.Azim@mau.edu.eg (M.A.M.); melkholy@aiu.edu.eg (M.E.)
*Corresponding author

***Abstract*—Intelligent transportation systems require automatic traffic flow prediction and vehicle speed estimation from images or videos. However significant occlusions, variations in vehicle appearance, and differing camera perspectives complicate these tasks. Addressing these challenges requires robust algorithms for vehicle detection and tracking that can adapt to changes in car orientation and illumination. In this work, we propose a novel approach that combines modern deep learning models with classical computer vision techniques to enhance both detection and tracking accuracy. We begin by improving the baseline performance of Faster Region-based Convolutional Neural Network (R-CNN) through strategic modifications, leading to reduced computational complexity and enhanced detection accuracy. To further improve tracking, we integrate Linear Constrained Tracking (LCT) with the Deep Simple Online and Realtime Tracking (DeepSORT) algorithm, effectively minimizing false positives caused by unreliable detections. Additionally, we address the challenge of transitioning from image space to the real world by estimating rectifying transformations using vanishing points, which significantly improves vehicle speed estimation. The results of our experiments show that our suggested Improved R-CNN_DeepSORT framework improves both accuracy and performance by a large amount, beating out other methods in the field.**

***Keywords*—traffic flow prediction, vehicle speed estimation, deep learning, faster Region-based Convolutional Neural Network (R-CNN), Deep Simple Online and Realtime Tracking (DeepSORT), image-to-real-world transformation, vehicle tracking**

I. Introduction

Traffic congestion, accidents, and pollution have increased due to the growing number of vehicles on the road, straining infrastructure. Because these issues affect daily life, traffic management systems must be robust and effective. As urbanization expands, the need for creative solutions grows, especially given the massive daily traffic data. This data covers vehicle movement, distribution, trends, and accidents, thereby aiding in the resolution of traffic concerns. Traditional traffic management has used statistical analysis to assess and decrease collision risks, congestion, and environmental implications. Traffic crash data, for example, can reveal links between traffic volume, collision frequency, and severity. Such data aids studies on fuel usage and pollution reduction, making transportation systems more sustainable [1]. Furthermore, Google Maps may not capture real-world variability, limiting distance measurements and journey estimations. It delivers convenient and reliable information, although map data precision, traffic circumstances, elevation variations, and Global Positioning System (GPS) accuracy can cause inconsistencies [2]. Average trip time statistics can be confusing because they use generic data that may not apply to all cases. It's vital to remember that these estimations are approximations and may not match reality.One of the purposes of the presented taxonomy is to provide a foundation for comprehending the primary components and concerns at a high level, as illustrated in Fig. 1. The taxonomy delineates the speed estimation issue in its entirety, encompassing all its constituent elements, methodologies, and experimental intricacies Given the widespread embrace of deep learning and technology for autonomous driving growing, active research areas in intelligent traffic analytics, such as vehicle speed estimation, anomalous event detection on streets (including traffic crashes), and vehicle verification, have emerged [3].

Recent advancements in technology have introduced deep learning and computer vision as powerful tools for improving traffic management.There are two types of deep learning-based object detection: One-stage (e.g., You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD) and two-stage (e.g., Fast-RCNN, Faster-RCNN) object detection methods based on deep learning have revolutionized the detection and tracking of vehicles [4, 5]. Using the targeted search approach, R-CNN retrieves about 2000 area suggestions from the top to the bottom of the image. Then, features are retrieved from each region suggestion and each region suggestion is categorized using SVM. Finally, each region's proposal is subjected to

Manuscript received June 21, 2024; revised August 14, 2024; accepted October 2, 2024; published February 10, 2025.

boundary regression. Convolutional layers for RPN and Fast R-CNN are also shared by the Faster RCNN in order to streamline operations and speed up computation [6]. Full Convolutional Neural Networks (FCNN) have further extended this integration for specific tasks like car taillight recognition, showcasing the adaptability of these models to various aspects of traffic monitoring [7].

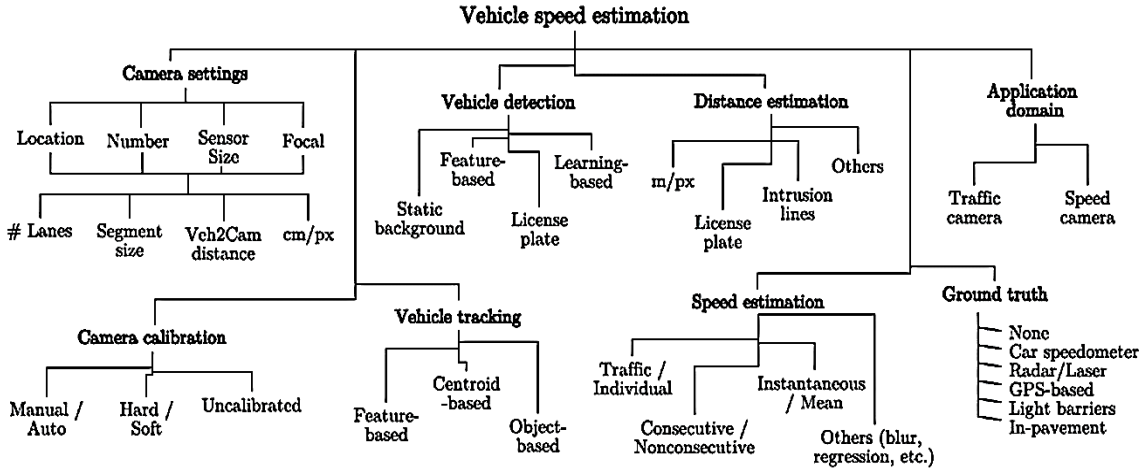


Fig. 1. Vehicle speed estimation taxonomy [8].

Following the process of object detection, the subsequent step involves the utilization of tracking techniques to ascertain the precise spatial coordinates of the recognized items in every frame. The inputs for the tracking process consist of objects that are identified and enclosed within bounding boxes. In a comparable way, tracking several objects is more difficult than tracking just one object. For monitoring several objects, the Kalmal filter has been used in earlier works [9, 10]. The efficiency of multiple object tracking with initialization and learning processes was also examined using the particle filter method [11]. The DeepSORT(Simple Online and Real time Tracking) algorithm, Currently, a widely utilized technique for object tracking incorporates the integration of several components, namely the Kalman filter, the Mahalanobis distance, the Hungarian algorithm, and the appearance feature vector [12]. The performance of trackers is being severely restricted by false positive tracks brought on by suspect detection outcomes [3]. In this paper, In addition to the main Deep Simple Online and Realtime Tracking (SORT) tracker, we're introducing low confidence track filtering to the main Deep SORT tracker to more effectively handle this problem.

Finally, the most difficult challenge in estimating vehicle speed is modeling the transition from the image domain to the actual world such that the velocity of the vehicles can be inferred from samples gathered in the image domain. In light of these challenges, this paper proposes a novel approach to enhance traffic flow estimation and vehicle speed prediction by combining deep learning models with classic computer vision techniques. Our contributions are threefold:

- **We are developing an innovative traffic flow estimation system.** Leveraging the latest advancements in vehicle detection, tracking, and speed estimation, our system addresses the limitations of existing methods by improving accuracy and efficiency.
- **Implementing low-confidence track filtering in DeepSORT:** Using Long-term Correlation Tracker Long-term Correlation Tracker (LCT) filtering cuts down on false-positive tracks by a large amount, making multi-object tracking more reliable in real-time traffic situations.
- **Modeling the transition from image space to real-world speed estimation:** To address the critical challenge of vehicle speed estimation, we implement a rectifying transformation using vanishing points in the image, ensuring accurate velocity predictions.

Our comprehensive evaluation demonstrates that the proposed system not only achieves high accuracy but also shows potential as a foundation for future high-level traffic management models. The paper is organized in the following structure. The "Related works" section provides a comprehensive evaluation of pertinent studies. The "Methodology" section outlines our approach to the challenge of estimating traffic flow, examines the design decisions we made, and presents the dataset we obtained. The "Experiments" section provides the evaluation procedure and implementation specifics of the experiment. The "Results and Discussions" section presents a detailed analysis and overview of the data that were gathered. The "Conclusions" section serves as the final part of the study, summarizing the main findings and suggesting potential areas for further research.

II. Related Work

Numerous studies have been conducted in the past regarding the estimation of vehicle speed. These studies can be categorized into two types based on their implementation: those that utilize depth information to compute vehicle speed, and those that utilize reality information displayed on the screen to do so. Additionally, due to advancements in deep learning and the rapid

enhancement of computer capabilities, numerous innovative traffic state estimating techniques have emerged. The tensor decomposition based model utilizes tensor decomposition and incorporates neighborhood information as a graph regularization technique for traffic state estimate [13]. Tensor decomposition approaches typically structure traffic status data as a third-order or higher tensor, such as road segment × day × time interval. Computing these high-dimensional tensors demands significant memory and computational resources. Du *et al.* [14] utilized LSTM NN and Convolutional Neural Network (CNN) to predict highway traffic speed while accounting for numerous traffic flow parameters. Nevertheless, the aforementioned analysis neglected to account for the spatial correlations of road network traffic conditions. In 2019, Lei *et al.* [15] developed a speed prediction system based on stereoscopic vision, which utilizes a pair of cameras for calibration and subsequently establishes a binocular stereoscopic vision system. The methodology entails the identification and comparison of identical vehicles present in two calibrated images, followed by the determination of their spatial coordinates and the extraction of their distinctive features. The determination of the vehicle's depth is achieved by computing the disparity between the two images, so obtaining the three-dimensional coordinates of the vehicle. Ultimately, by utilizing the 3D coordinates and the number of video frames, one may ascertain the velocity of the vehicle. Hartoto [16] utilized image processing to compute vehicle speed through IP-Camera Traffic and the Foreground Extraction method. At 80 km/h, the algorithm achieved an unprecedented 93% accuracy rate. In 2016, Luvizon *et al.* [17] suggested a road calibration technique that involves using perspective transformation. This method entails placing a 4.8 m × 2.0 m rectangle on the road and determining its pixel position in the image. The calibration matrix is then calculated using perspective transformation based on these two sets of information. Video interpolation is utilized to track moving vehicles and determine their speed by analyzing the number of video frames and the distance traveled. This approach necessitates measuring the road first and driving the car from the bottom to the top of the image inside the designated region.

Viktor Kocur suggests using perspective transformations to identify 3D vehicle boundaries for speed estimation [18]. For this strategy to be effective, it is only necessary to position the camera on a level and straight road, with no significant additional location prerequisites. By employing perspective transformation, it is possible to alter the four vanishing points of the road in such a way that they resemble the original road shape. This can be achieved by the utilization of image processing techniques, which enable the identification of the vanishing points of the road within the given image. In order to build the calibration matrix, it is necessary to calibrate the road using the chosen scale, as it is impractical to fully restore the road's original dimensions. The subsequent procedure involves doing an examination of the vehicle utilizing machine learning techniques, generating a three-dimensional boundary framework for the vehicle, and determining its speed by considering the distance covered and the number of video frames captured. In conclusion, we have enhanced the previously mentioned method to overcome some constraints. Our suggested technique utilizes deep learning to detect vehicle objects without being restricted by the movement direction. It determines the size of all road markers according to traffic marking regulations and calculates the calibration matrix using perspective transformation based on the road mark size. Thus, measuring the road size or determining the image scale in advance is unnecessary for calibrating and completing the speed calculation [3]. Our suggested system consists of modules for speed estimates, robust tracking, and multi-object identification. The primary contribution of this study is the suggestion of an automated vehicle speed estimation technique that concerns operational speed without requiring expensive procedures like camera calibration.

III. Methodology

The goal of this work is to create a method for estimating traffic flow that involves counting and the classification of vehicles according to their directional movement. To do this, the issue is divided into three subtasks: vehicle detection, vehicle tracking, and vehicle speed estimation, as seen in Fig. 2. It logically results in a modular and testable design comprised of detection and tracking components. Each module is described in depth in the following sections, along with the data collected for training and evaluation.

A. Vehicle Detection

Convolutional Neural Networks (CNN) provide the foundation for the majority of object detectors now in use. These detectors are divided into two types: single-stage detectors and two-stage detectors. The Single-stage detectors are often quick and anticipate classes and objects bounding boxes during a single network function. These designs perform especially well when the target items take up a sizable portion of the image. Two-stage detectors, as opposed to single-stage detectors, predicted areas in the first stage before enhancing and classifying each one in the second. An easy method was used in early R-CNN [19] where areas were created an algorithm for selective search and then input into the classification CNN. The R-CNN's overall performance is low because of the restricted compute time and the necessity to run a large classifier per area. Fast R-CNN was proposed to overcome this constraint [20]. Faster R-CNN uses a Region Proposal Network (RPN) that significantly reduces the proposal cost by sharing convolutional layers within the object identification network. At each location, an additional few convolutional layers are used to regress region boundaries and rates for abjectness. While this algorithmic design tweak resulted in considerable speed improvements, it also improved object detection performance [21]. The R-CNN's overall performance is low because of the restricted compute time and the necessity to run a large classifier per area. Fast R-CNN was proposed to overcome

this constraint [20]. A Region Proposal Network (RPN) in Faster R-CNN shares convolutional layers with the object detection network, considerably lowering the proposal cost as shown in Fig. 3. A few more convolutional layers are utilized at each site to regress region borders and abjectness ratings. While this algorithmic design tweak resulted in considerable speed improvements, it also improved object detection performance. The main component of our detection module is the commonly used two-stage Faster R-CNN [22] detector, which performs CNN once for each region instead. Two factors led to this decision: First one, two-stage detectors offer a reliable starting point for every detection job and are now state-of-the-art in many detection benchmarks [23, 24]. Second, adding a new prediction branch is all that is required to expand the faster R-CNN architecture to accommodate multiple workloads.

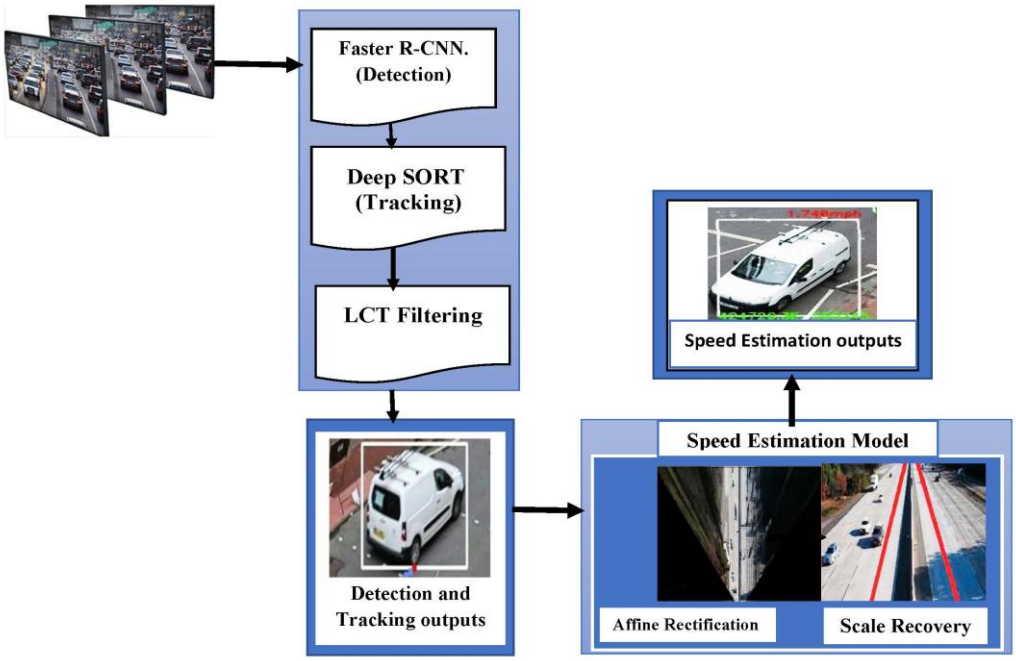


Fig. 2. Processing workflow.

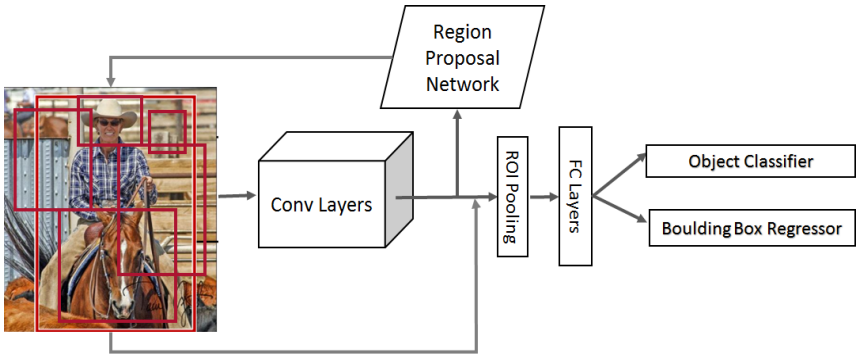


Fig. 3. The system architecture of faster R-CNN [25].

B. *Vehicle Tracking*

1) *Deep SORT*

The Sort algorithm employs a basic Kalman filter to analyse the correlation of frame-by-frame data and the Hungarian algorithm to ascertain their connection [20, 21]. High frame rates work nicely with this simple technique [22, 23]. Sort, however, can only be precise if the object state estimation error is modest since it ignores the surface characteristics of the object to be discovered [24]. CNN is trained on a sizable pedestrian data set to extract features using Deep SORT, which substitutes the association measure with a more reliable measurement. This increases the network's resilience to loss and blockages [25, 26]. The association measure is replaced with a more reliable metric in Deep SORT, and CNN uses training on a large pedestrian data set to derive features, boosting the network's resilience to loss and impediments. The state estimate of the Deep SORT technique represents each of the location, aspect ratio, height of the centre of the bounding box, and associated speed data as an 8-dimensional space. A Kalman filter is then used to estimate the update trajectory [27]. The Kalman filter utilises a linear observation model and a uniform model as its observation variables.

Each track has a threshold that is used to keep track of how long it has been since the last successful pairing up to the present [28]. When the value surpasses the predetermined threshold Amax, the track is regarded as having ended. Deep SORT offers two measures for motion and appearance data to handle the assignment problem. By comparing the squared Mahalanobis distance between expected Kalman states and recently received observations, Eq. (1) demonstrates how to incorporate motion information:

$$d^{(1)}(i,j) = \left(d_{j-}y_i\right)^T S_i^{-1}\left(d_{j-}y_i\right) \quad (1)$$

where $\left(y_i, S_i\right)$ stands for the projection of the $j-th$ track distribution into measurement space and (d_j) stands for the $j-th$ bounding box detection. The next metric, as shown in Eq. (2), displays visual data between the $i-th$ track and the $j-th$ detection using the smallest cosine distance:

$$d^{(2)}(i,j) = min\left\{1 - r_J^T r_K^{T(i)} \middle| r_K^{T(i)} \in R_i\right\} \quad (2)$$

where r Is the appearance descriptor and R_i is the appearances of the last 100 object associated with the $i-th$ track. Where the appearance descriptor $\boldsymbol{r_j}$ is computed for each bounding box with $\|r_j\| = 1$.Additionally, a gallery $\left\{r_K^{T(i)}\right\}_{k=1}^{L_k}$ is maintained, where L_k represents the final number of linked each track's visual descriptions K. The association problem is resolved by utilizing a weighted sum of the two metrics, as illustrated in Eq. (3):

$$C_{i,j=\lambda d^{(1)}(i,j)+(1-\lambda)d^{(2)}(i,j)} \quad (3)$$

The more often seen objects might be given a greater priority thanks to Deep SORT's cascade matching method. This allows the trajectory with the same occlusion time to be thought of as allocating every time. After all, if the car is successfully tracked, Deep SORT would show the white tracking boxes, a vehicle's ID, and the center point coordinates (x, y).

2) Low Confidence Track (LCT) filtering

False positive tracks created by untrustworthy detection outcomes have a negative impact on tracker efficiency [29, 30]. To address this issue more effectively, we suggest a LCT filtering migrate to the basic DeepSORT tracker. Besides from the detection confidence threshold t_d, we compute the mean detection confidence of new detections linked with the tentative track in (*f*+1), (*f*+2), … (*f*+t_Tentative) frames in our extension. The Tentative track may only be changed to Confirmed track if the average detection confidence exceeds our predetermined average detection confidence level threshold t_aved. The tentative track gets removed in such a case. In this method, the detection results are filtered in two steps using both a t_d and a t_aved threshold rather than just a single t_d threshold. In order to prevent missed detections, td can be lowered to a lower value, and t_avedcan aid in suppressing the false positive tracks caused by low t_d. The detailed methodology of expanded low confidence track filtering for tentative tracks is presented in Low Confidence Track algorithm [26].

Algorithm 1: Low Confidence Track algorithm

Input: -Tentative tracks T_t
-Tentative threshold $\mathrm{t_{tentative}}$
-Average detection confidence threshold $\mathrm{t_{aved}}$
-Associated detection confidence p_t

Output: - Confirmed tracks T_c
- Deleted tracks T_d

Steps:

For sequential frames do
For $t \in \mathrm{T_t}$ do
If t is new in $T_t$then
$hits = 0$
$total\_prob = 0$
$hits = hits + 1$
$total\_prob = total\_prob + p_t$
$if\ \frac{total\_prob}{hits} < \mathrm{t_{aved}}$
$T_d = T_d \cup t$ and $T_t = T_t \setminus \mathrm{t}$
$Else$
$T_c = T_c \cup t$ and $T_t = T_t \setminus \mathrm{t}$
End of Steps

C. Speed Estimation

The speed estimation method is divided into two sections: (1) affine rectification, which restores the scene's affine-invariant properties, and (2) scale recovery, that estimates vehicle speed in the actual world. These steps are performed after the pixel velocities have been collected in the image domain, using the Kalman filter.

1) Affine rectification

Uncelebrated cameras take the traffic recordings. As a result, each frame in the video is an attempt to project the 3D world onto the plane of the image. Because projections transformations do not preserve ratios between segments, it is difficult to estimate speed in the image domain directly. In this study, we make the assumption that the majority of the roadways can be accurately represented as a flat surface. The rectification technique is applied to these planar sections. which estimate a homographs H that maps points $x = [x, y, 1]^T$ in the image domain to points $X = [X, Y, 1]^T$ in the rectified domain [27] where x and y represent the horizontal and vertical pixel location of the center of the bottom two coordinates of the bounding box.

$$H_X = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = X \quad (4)$$

Speed estimate might be compromised in non-planar places, but the scale recovery described in next section will make up for this [31]. We note that once H is known, both sides of Eq. (3) may be differentiated to yield the velocity in the corrected domain.

$$\begin{bmatrix} X^* \\ Y^* \end{bmatrix} = \begin{bmatrix} \frac{h_{11}+C_{2,3}y}{(h_{31}x+h_{32}y+1)^2} & \frac{h_{12}+C_{2,3}x}{(h_{31}x+h_{32}y+1)^2} \\ \frac{h_{21}+C_{1,3}y}{(h_{31}x+h_{32}y+1)^2} & \frac{h_{11}+C_{1,3}x}{(h_{31}x+h_{32}y+1)^2} \end{bmatrix} \begin{bmatrix} x^* \\ y^* \end{bmatrix}, \quad (5)$$

where $h_{i,j}$corresponds to the minor of H represented by $C_{i,j}$. One traditional method of determining H is based on finding the sites where two perpendicular directions converge.

In homogeneity coordinates, we designate the two point of vanishing as $v1$ and $v2$, where $v1$ denotes the road axis's direction and $v2$ denotes the direction perpendicular to $v1$. The following equations are thus valid:

$$Hv_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, Hv_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad (6)$$

where H from:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ - & v_1 \times v_{21} & - \end{bmatrix} \quad (7)$$

You may use Eqs. (6) and (7) to solve the parameters in Eq. (4). By choosing certain spots in the image, we manually identify two vanishing points for each place. We set $v2 = [0, 1, 0]^T$ because Parallel lines to the roadway axis are nearly identical in the image domain for locations 1, 2, and 3.

2) Scale recovery

Converting speed from the pixel space to the actual world domain, it is crucial to restore the scale after picture rectification. Both horizontal and vertical directions need to be addressed during this rehabilitation [32]. For scale recovery in the horizontal and vertical dimensions, we utilize the length of the road white strips and the width of the lanes, respectively. We estimated the distances in the actual world using a rough estimate from Google Maps [33]. Road lanes become parallel after rectification, which guarantees a consistent lane width in the x direction in flat terrain. As a result, the following is our selection for the scaling factor in the x direction:

$$s_x = \frac{W}{w} \quad (8)$$

W Stands for actual lane width, expressed in meters and w stands for pixels in Eq. (8). There is a different strategy for the vertical direction. The pixels in the vertical direction stretch after projection and rectification, and this effect is particularly pronounced in the pixels close to the detected vanishing point. As a result, there is a nonlinear shift in the scale along the y direction. We used a linear compensatory to account for this scale fluctuation as follows:

$$s(y) = \frac{s_2 - s_1}{y_{max} - y_{min}}\, y + s_1, \quad (9)$$

$$s_{1} = \frac{L_1}{l_1} \quad (10)$$

$$s_{2} = \frac{L_2}{l_2} \quad (11)$$

where $L1$ and $L2$ are the lengths in metres of the two white stripes at the two different heights. The equivalent lengths in pixels in the converted image are $L1$ and $L2$. Finally, the vehicle's estimated speed is given by:

$$\sqrt{(s_x X^*)^2 + \left(s_y Y^*\right)^2} \quad (12)$$

With heights varying from y_{min} to y_{max} in the image's space, all of the vehicles in the window have their speeds estimated.

IV. Experimental Results

A. Datasets

1) KITTI Dataset

A popular reference for vehicle detection and estimation systems is KITTI [21].

a) Dataset overview

- The KITTI dataset is designed for various computer vision tasks, including vehicle detection, object tracking, depth estimation, and visual odometer.
- For vehicle detection specifically, the dataset contains:
  - **7518 images for testing**: These images are used to evaluate the performance of the model. They come with ground truth annotations for object detection tasks but are often kept hidden in official leaderboards to prevent overfitting on test data.
  - **7481 images for training**: These images are used to train the models and come with ground truth labels, including the bounding boxes for vehicles and other objects.

b) Difficulty categories

a) The KITTI dataset categorizes vehicles into three difficulty levels to provide a more nuanced evaluation of detection algorithms. These categories are based on three factors: **size, occlusion**, and **truncation**.

b) **Easy**: Vehicles that are larger, fully visible (or nearly so), and not truncated. These are typically vehicles close to the camera with minimal obstructions.

c) **Moderate**: Vehicles that are medium-sized, partially occluded, or partially truncated. These represent a more challenging detection scenario due to obstructions or reduced visibility.

d) **Hard**: Vehicles that are small, heavily occluded, or highly truncated. These are the most challenging to detect, often due to distance from the camera, significant occlusions, or being partially out of frame.

By categorizing the vehicles in this manner, the KITTI dataset allows researchers to evaluate the performance of detection algorithms across varying levels of difficulty, making it a valuable benchmark for advancing vehicle detection and estimation technologies [28]. This structured approach helps in assessing the robustness and generalizability of different methods under real-world conditions.

2) NVIDIA AI city challenge dataset

The AI City Challenge was established by NVIDIA with the aim of fostering the adoption of advanced video analysis techniques to enhance the intelligence and safety of urban environments [29, 34]. It comprises 27 video clips captured by stationary cameras at 4 distinct sites in California, U.S.A. Each clip lasts for 1 minute and contains a total of 1800 frames. The traffic flow on I-280 highway is captured at the first two places, each with 16 clips. The video clip only shows two opposing driving directions along the vertical axis. Each vehicle maintains a consistent

pace without changing direction, such as driving left, right, or making a U-turn, due to the absence of intersections or traffic signals. These two locations are perfect for applying our suggested approach to calculate velocity using landmark-based scan lines. The video clips from the final two sites, each including 11 clips, provide traffic statistics at junctions with traffic lights in metropolitan areas.

3) DAWN

This dataset comprises a diverse collection of genuine photos captured under various demanding weather conditions. We collect the photographs from various traffic environments such as metropolitan areas, highways, and freeways, which display a wide range of traffic patterns. We have categorized a total of 1000 photos into four distinct weather conditions: fog, snow, rain, and sandstorms. The holdout methodology partitions the 1000 picture dataset into training and testing sets in a 3:1 ratio. The training and testing sets provide a balanced distribution of each weather scenario [30].

B. Evaluation Metric

The effectiveness of the Faster R-CNN detection architecture with RPNs has been assessed using several metrics, specifically accuracy, precision, and recall. These metrics are calculated as follows:

$$recall = \frac{TP}{TP + FN} \tag{13}$$

$$percision = \frac{TP}{TP + FP} \tag{14}$$

TP, *TN*, *FP*, and *FN* reflect the concepts of true positives, true negatives, false positives, and false negatives, respectively.

$$\text{accuracy rate} = \frac{\text{recall} + \text{percision}}{2} \times 100 \tag{15}$$

1) Depth evaluation metrics

We modify the depth evaluation criteria by transitioning from image-level to object-level assessment, using both error and accuracy measurements [31] to enhances the precision and relevance of depth estimation in complex scenes, such as those encountered in autonomous driving and computer vision tasks.

a) Error Metrics

- **Absolute Relative Difference (Abs Rel)**: This metric calculates the mean of the absolute differences between the predicted depth $y_i$and the ground truth depth y_i^*normalized by the ground truth depth as N is the total number of depth values being considered

$$Abs\ Rel = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - y_i^*|}{y_i^*} \tag{16}$$

This metric helps in understanding how much, on average, the predicted depth deviates from the actual depth relative to the ground truth. Where N is the total number of depth values being considered.

- **quared Relative Difference (Sq Rel):** Similar to Abs Rel, this metric calculates the squared differences between the predicted and ground truth depths, normalized by the ground truth depth. It is defined as:

$$Sq\ Rel = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - y_i^*)^2}{y_i^*} \tag{17}$$

Sq Rel emphasizes larger errors more than Abs Rel, providing a sense of how severe the errors are in the depth estimation.

- **Root Mean Square Error (RMSE)**: This metric calculates the square root of the average squared differences between the predicted and ground truth depths:

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2} \tag{18}$$

RMSE provides a single measure of the error magnitude and is sensitive to large errors due to squaring, making it useful for evaluating overall accuracy.

- **RMSE Log**: The logarithmic version of RMSE, which calculates the error in logarithmic space:

$$\boldsymbol{RMSE\ Log} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log y_i - \log y_i^*)^2} \tag{19}$$

RMSE Log is less sensitive to large errors and is particularly useful when the depth range varies widely.

b) Accuracy metrics

The accuracy metrics assess the percentage of depth predictions that fall within a certain threshold of the ground truth. The condition is defined as:

$$\max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) < \delta \tag{20}$$

where $takes\ values = 1.25,\ 1.25^2,\ 1.25^3$. These thresholds allow for different levels of acceptable error $\delta = 1.25$ Evaluates the percentage of predictions that are within 25% of the ground truth. $\delta = 1.25^2, 1.25^3$ these higher thresholds evaluate the percentage of predictions within 56.25% and 95.3125% of the ground truth, respectively. They provide a broader view of how well the model performs under less strict criteria.

2) Orientation assessment

Orientation Score (OS): The orientation score is a metric used to evaluate the alignment or orientation of predicted objects compared to their ground truth counterparts. In accordance with the KITTI framework, OS is often used in the context of 3D object detection and depth estimation tasks. It measures how well the predicted orientation of an object matches the actual orientation, which is crucial for applications like autonomous driving, where the correct orientation of vehicles and pedestrians directly impacts safety and navigation decisions [32].

C. Vehicle Detection, Vehicle Tracking, and Speed Estimation Evaluation

1) DAWN dataset test

We have assessed the effectiveness of the faster R-CNN detection architecture with Region Proposal Networks

(RPNs) using measures like accuracy, precision, and recall. We obtain the metrics mentioned in Table I by computing the intersection over union (IoU) between the detected bounding boxes and the ground-truth bounding boxes. Instances that have an Intersection over Union (IoU) value greater than 0.8 are classified as true positives. The proposed method accurately detects on-road cars in test photos from the DAWN dataset, which consists of four weather conditions: fog, snow, rain, and sandstorms. This is demonstrated in Fig. 4. Red boxes indicate the detected automobiles.

TABLE I. RESULTS OF VEHICLE DETECTION EMPLOYING THE TRADITIONAL FASTER R-CNN WITH RPN

Weather condition	Accuracy (%)	Precision (%)	Recall (%)
Fog	91	92.1	91.55
Snow	90.5	90.7	90.6
Rain	91	91.3	91.15
Sandstorms	90	90.1	90.05

Fig. 4. Accurate identification of on-road vehicles utilizing the proposed approach on test images sourced from the DAWN dataset under fog, snow, rain and sandstorm conditions.

Various vehicle detection algorithms have been evaluated using the public DAWN dataset in this study, and their results have been compared to those of the proposed system. The comparisons are presented in Table II for DAWN datasets, using average precision as the criterion for comparison. The average precision is calculated by combining the individual precisions obtained from test samples under various weather circumstances. This investigation confirms that the suggested detection system accurately identifies vehicles on the road in both ideal (sunny) and difficult weather situations.

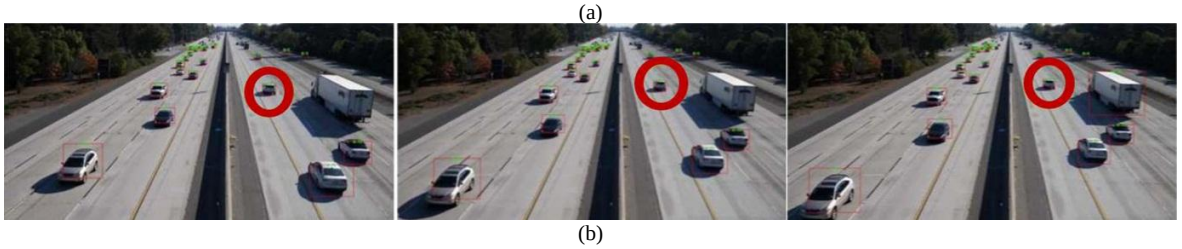
TABLE II. DAWN DATASET COMPARATIVE PERFORMANCE STUDY WITH PREVIOUSLY RELEASED RESEARCH

Reference	Backbone	Ap (%)	Processing speed (Fps)
Hu *et al.* [33]	VGGNet-16	79.2	17
Hassaballah *et al.* [34]	NA	82.5	18
Redmon *et al.* [35]	DarkNet-53	88.6	24
Li *et al.* [36]	ResNet-101	75.43	18
Ghosh [37]	Faster R-CNN	89.5	20
proposed	Faster R-CNN+ RPN	92.3	20

2) NVIDIA AI city challenge dataset test

We analyze and contrast the tracking accuracy of SORT and DeepSORT in Fig. 5 and Table III. Fig. 5 demonstrates that DeepSORT, with the aid of the acquired deep representation, produces more reliable and consistent tracking results compared to SORT. SORT relies solely on bounding box data for association and tends to lose track when the detections of the vehicle being tracked are absent for multiple consecutive frames. This shows that a tracking approach utilizing a similarity model may produce consistent tracking outcomes with reduced occurrences of identity shifts. Fig. 4 displays this impact. Quantitative data from the assessment server indicate that the Root Mean Square Error (RMSE) is marginally reduced when utilizing DeepSORT as a tracking method compared to SORT. We suggest that both methods may achieve equivalent detection rates by localizing a vehicle in 30% of frames over the whole movie, given appropriate hyper-parameter configurations. If a vehicle is monitored for an extended period, inaccuracies in speed estimation are more likely to occur due to the assessment metric's design.

(a)



(b)

Fig. 5. An analysis of the tracking findings acquired from SORT and DeepSORT. (a) In the SORT algorithm, a track becomes lost and is then reinitialized after a few frames; (b) DeepSORT demonstrates efficient vehicle tracking capabilities without losing track.

TABLE III. SPEED ESTIMATE ON THE NVIDIA AI CITY CHALLENGE DATASET UTILIZING SEVERAL TRACKING TECHNIQUES IS COMPARED

Method	RMSE
SORT	2.231
Deep SORT	1.168

Table IV displays the identification rate and velocity inaccuracy of our proposed technique using the NVIDIA AI City Challenge Dataset. The initial two sites containing 15 out of 26 video clips are optimal scenarios for applying our technique to calculate velocity. All vehicles are moving steadily without any notable changes in speed or direction. Our technique in this situation results in a Root Mean Square Error (RSME) of 5.9862 mph. For the second scenario, we analyze the intersections in metropolitan regions using 11 out of 27 clips. Our technology accurately predicts velocity when vehicles consistently travel down a roadway and intersect virtual scan lines. At the intersection of two roads, the landmark patterns may be incomplete, causing our approach to be unable to properly assess the velocity from the last scan line to the traffic light, especially in the middle of the crossroad. The RMSE for this scenario is 10.8039 mph, with errors concentrated around the junction of two roadways during vehicle deceleration or acceleration. We analyze all four sites using 26 video clips in the last two instances. By applying refinement approaches as outlined in Section IV.A.1), we reduced the RMSE to 7.8143 mph for case 4, compared to 8.3600 mph without refinement for case 3.

TABLE IV. EVALUATION OF VELOCITY IN DIFFERENT LOCATIONS USING NVIDIA AI CITY CHALLENGE DATASET

Case	Locations	#Clips	DR	RSME
1	1,2	15	1	5.9862
2	3,4	11	1	10.8039
3	1,2,3,4	26	1	8.3600
4	1,2,3,4	26	1	7.8143

3) KITTI dataset test

Table V shows that additionally, faster R-CNN was assessed at several training scales between 800 and 1800. As predicted, as the test scale grows, the performance of Faster R-CNN improves steadily across all three categories, and the trend does not appear to have stopped until 1800. This means that performance may be improved beyond a scale of 1800. We chose average accuracy (AP) as our performance statistic and computed it with the tool.

TABLE V. AVERAGE PRECISION (AP) OF FASTER R-CNN AT VARIOUS TRAINING SCALES USING THE KITTI DATASET

Training scale	Easy/AP	Moderate/AP	Hard/AP
1800	0.98	0.81	0.7
1500	**0.98**	**0.8**	**0.65**
1200	**0.88**	**0.69**	**0.6**
1000	**0.82**	**0.62**	**0.55**
800	**0.78**	**0.57**	**0.45**

Fig. 6 displays instances of both successful and unsuccessful results obtained by our approach on the KITTI dataset. The KITTI dataset uses the color red to represent genuine detections, blue to represent missing detections, and green to represent incorrect detections.

Fig. 6. Examples of both successful and unsuccessful cases.

The evaluation in Table VI showcases the efficacy of our model across various individual metrics. The models undergo training and testing using the KITTI dataset, which is of the same nature. When varying the amount of training data, the findings indicate that a data-hungry network benefits from having a larger data capacity, leading to improved performance.

TABLE VI. PERFORMANCE OF SPEED ESTIMATION

Amount %	Dataset	Depth Error Metrics				Depth accuracy Metrics			Orientation
		RMSE log	*RMSE*	*Sq Rel*	*Abs Rel*	$\delta < 1.25^3$	$\delta < 1.25^2$	$\delta < 1.25$	(OS)
KITTI	0.232	10.168	5.212	0.249	1	0.822	0.922	0.943	0.742
	0.134	5.971	4.651	0.213	10	0.865	0.955	0.941	0.880
	0.125	5.245	4.234	0.124	30	0.944	0.964	0.859	0.882
	0.102	4.484	3.361	0.112	100	0.964	0.960	0.953	0.923

TABLE VII. KITTI BENCHMARK RESULTS ALL METHODS ARE RANKED ON THE "MODERATE" SCALE

Model	AP (%)			Runtime
	Moderate	Easy	Hard	
CASDC [38]	96.13 %	98.66 %	93.26 %	0.1 s
VirConv-S [39]	97.27 %	98.00 %	94.53 %	0.09 s
GraR-VoI [40]	96.38 %	96.81 %	91.20 %	0.07 s
SFD [41]	96.17 %	98.97 %	91.13 %	0.1 s
VPFNet [42]	96.15 %	96.64 %	91.14 %	0.06 s
proposed	97.12%	98.21	95.00	0.01

Table VII represents a Comparative Analysis with State-of-the-Art. All of the training set is used to train our models. and we compare them to the other six approaches that have been published. With the exception of VirConv-S [39], which is sluggish but obtains the best accuracy on a moderate example, it is evident that The approach suggests demonstrating superior accuracy and efficiency in terms of speed. Reducing the size of the input images allows us to use our technique to attain the more

accuracy than VirConv-S. In the comparative analysis, while our approach demonstrated superior accuracy and efficiency compared to six state-of-the-art methods, including VirConv-S [39], Although reducing the size of the input images helped us achieve greater accuracy than VirConv-S, this approach may introduce potential faults. Smaller image sizes could lead to a loss of critical details, particularly in complex scenes or with individuals having less distinct facial features. This can result in increased false negatives or decreased robustness in face recognition tasks. Even if we do not reduce the size of the input images, our algorithm remains robust and competitive. Reducing image size was one optimization strategy that allowed us to achieve greater accuracy compared to VirConv-S. However, our algorithm maintains strong performance across various scenarios without this reduction. The key strengths of our approach, including its accuracy and efficiency, are preserved even when using standard image sizes. This demonstrates that our method is not only effective with optimized image sizes but also performs reliably in its default configuration, making it a robust solution compared to other state-of-the-art methods.

V. Conclusion

In this study, we provide an enhanced R-CNN_DeepSORT architecture that addresses three shortcomings of the original R-CNN_DeepSORT design and allows for quick vehicle detection and speed estimation with a wide range of scales. The first contribution, Faster-RCNN, enhanced R-RCNN by generating region proposals using Region Proposal Network (RPN) rather than the original selective search approach in order to decrease operations and boost performance. The tracking process is used to determine the locations of the detected objects in each frame, which is the second contribution. We suggest using a Deep SORT algorithm with simple online and real-time tracking to add low-confidence track filtering. The operating speed is the subject of this paper's third contribution. To reduce False Positive Tracks brought on by Unreliable Detection Results That Affect the Performance of Trackers, the hardest challenge in determining a vehicle's speed estimate is to characterize the transition from the image domain to the actual world. We initially estimate a rectifying transformation utilizing the image's vanishing points in order to retrieve the affine properties. Then, the difference in scale between the image's domain and the actual world is determined. The R-CNN Deep SORT Architecture demonstrates exceptional performance in terms of both accuracy and speed on the KITTI benchmark and NVIDIA AI City dataset, positioning it at the forefront of current advancements in this field. All of the training sets are used to train our models. When we compare them to the other six approaches that have been published the accuracy is 95 % on a hard example and runtime of 0.01 seconds which makes evident that the approach suggests demonstrating superior accuracy and efficiency in terms of speed. Additional research efforts could involve the assessment of the R-CNN_Deep SORT Architecture on more intricate datasets, as well as its integration into various intelligent transportation systems. For further work, analyze larger and more diversified datasets with traffic, weather, and nighttime data. Using LiDAR, radar, and thermal imaging could improve detection in difficult conditions. Transformer-based designs and self-supervised learning may increase performance. We should optimize the model for real-time deployment on edge devices and incorporate privacy-preserving mechanisms to ensure efficiency and ethical compliance in practical implementations.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Marwa A. Marzouk conceived and designed the analysis, analyzed the data, and wrote the paper. Mohamed Elkholy collected the data, contributed data and analysis tools, and performed the analysis. Both of the authors wrote the paper. All authors had approved the final version.

References

[1] P. H. Tobing, "Application of system monitoring and analysis of vehicle traffic on toll road," in *Proc. 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2014, pp. 1–5.

[2] M. A. Marzouk and M. Elkholy, "Combining bag of visual words-based features with CNN in image classification," *Journal of Intelligent Systems*, vol. 33, 20230054, 2024.

[3] M. A. Marzouk and A. Abd El Azeem, "Vehicles detection and counting based on internet of things technology and video processing techniques," *IAES International Journal of Artificial Intelligence*, vol. 11, 405, 2022.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[5] M. Elkholy and A. ElFatatry, "Framework for interaction between databases and microservice architecture," *IT Professional*, vol. 21, no. 5, pp. 57–63, 2019. doi: 10.1109/MITP.2018.2889268

[6] G. Zhong, Y.-H. Tsai, Y.-T. Chen, X. Mei, D. Prokhorov, M. James *et al.*, "Learning to tell brake lights with convolutional features," in *Proc. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1558–1563.

[7] M. I. Elkholy and M. A. Marzok, "Trusted microservices: A security framework for users' interaction with microservices applications," *Journal of Information Security and Cybercrimes Research (JISCR)*, vol. 5, no. 2, Dec. 2022.

[8] D. F. Llorca, A. H. Martínez, and I. G. Daza, "Vision-based vehicle speed estimation: A survey," *IET Intelligent Transport Systems*, vol. 15, pp. 987–1005, 2021.

[9] H.-K. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3d multi-object tracking for autonomous driving," arXiv preprint, arXiv:2001.05673, 2020.

[10] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *Proc. 2016 6th International Conference on System Engineering and Technology (ICSET)*, 2016, pp. 109–115.

[11] H. Supreeth and C. M. Patil, "Efficient multiple moving object detection and tracking using combined background subtraction and clustering," *Signal, Image and Video Processing*, vol. 12, pp. 1097–1105, 2018.

[12] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. 2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649.

[13] X. Chen, Z. He, and L. Sun, "A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation," *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 73–84, 2019.
[14] S. Du, T. Li, X. Gong, and S.-J. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning," arXiv preprint, arXiv:1803.02099, 2018.
[15] L. Yang, M. Li, X. Song, Z. Xiong, C. Hou, and B. Qu, "Vehicle speed measurement based on binocular stereovision system," *IEEE Access*, vol. 7, pp. 106628–106641, 2019.
[16] P. Hartoto, "Motor vehicle speed detection system in real time traffic information system," Thesis, Department of Electrical Engineering, Sepuluh Nopember Institute of Technology Surabaya, Surabaya, 2011. (in Indonesian)
[17] D. C. Luvizon, B. T. Nassu, and R. Minetto, "A video-based system for vehicle speed measurement in urban roadways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1393–1404, 2016.
[18] V. Kocur and M. Ftáčnik, "Detection of 3D bounding boxes of vehicles using perspective transformation for accurate speed measurement," *Machine Vision and Applications*, vol. 31, 62, 2020.
[19] T. Wang, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao, "Learning rich features at high-speed for single-shot object detection," in *Proc. the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1971–1980.
[20] R. Girshick, "Fast R-CNN," in *Proc. the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.
[21] M. Elkholy and M. A. Marzouk, "Deep learning-based classification of eye diseases using convolutional neural network for OCT images," *Frontiers in Computer Science*, vol. 5, 1252295, 2024.
[22] R. Faster, "Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 9199, pp. 2969239–2969250, 2015.
[23] M. Elkholy and A. ElFatatry, "Intelligent broker: A knowledge based approach for semantic web services discovery," in *Proc. the 10th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE-2015)*, Barcelona, Spain, 2015, pp. 39–44. doi: 10.5220/0005455300390044
[24] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi *et al.*, "A new benchmark and protocol for multi-object detection and tracking," arXiv preprint, arXiv:1511.04136, 2015.
[25] Q. Fan, L. Brown, and J. Smith, "A closer look at Faster R-CNN for vehicle detection," in *Proc. 2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 124–129.
[26] M. Elkholy and A. B. Mohamed, "Efficient security model for RDF files used in IoT applications," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 4, 2021. doi: 10.14569/IJACSA.2021.0120431
[27] A. Kumar, P. Khorramshahi, W.-A. Lin, P. Dhar, J.-C. Chen, and R. Chellappa, "A semi-automatic 2D solution for vehicle speed estimation from monocular videos," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 137–144.
[28] M. Elkholy, Y. Baghdadi, and M. Marzouk, "Snowball framework for web service composition in SOA applications," *International Journal of Advanced Computer Science and Applications*, vol. 13, 2022.
[29] H. Luo, W. Chen, X. Xu, J. Gu, Y. Zhang, C. Liu *et al.*, "An empirical study of vehicle re-identification on the AI city challenge," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4095–4102.
[30] M. A. Kenk and M. Hassaballah, "DAWN: Vehicle detection in adverse weather nature dataset," arXiv preprint, arXiv:2008.05402, 2020.
[31] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in Neural Information Processing Systems*, vol. 27, 2014.
[32] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 108–115.
[33] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin *et al.*, "SINet: A scale-insensitive convolutional neural network for fast vehicle detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 1010–1019, 2018.
[34] M. Hassaballah, M. A. Kenk, and I. M. El-Henawy, "Local binary pattern-based on-road vehicle detection in urban traffic scene," *Pattern Analysis and Applications*, vol. 23, pp. 1505–1521, 2020.
[35] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint, arXiv:1804.02767, 2018.
[36] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," in *Proc. the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6054–6063.
[37] R. Ghosh, "On-road vehicle detection in varying weather conditions using faster R-CNN with several region proposal networks," *Multimedia Tools and Applications*, vol. 80, pp. 25985–25999, 2021.
[38] Z. Yu, X. Li, H. Huang, W. Zheng, and L. Chen, "Cascade image matting with deformable graph refinement," in *Proc. the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7167–7176.
[39] H. Wu, C. Wen, S. Shi, X. Li, and C. Wang, "Virtual sparse convolution for multimodal 3D object detection," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21653–21662.
[40] Q. Xia, Y. Chen, G. Cai, G. Chen, D. Xie, J. Su *et al.*, "3-D HANet: A flexible 3-D heatmap auxiliary network for object detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–13, 2023.
[41] X. Wu, L. Peng, H. Yang, L. Xie, C. Huang, C. Deng *et al.*, "Sparse fuse dense: Towards high quality 3d detection with depth completion," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5418–5427.
[42] H. Zhu, J. Deng, Y. Zhang, J. Ji, Q. Mao, H. Li *et al.*, "VPFNET: Improving 3D object detection with virtual point based lidar and stereo data fusion," *IEEE Transactions on Multimedia*, 2022.