Expert Systems With Applications 227 (2023) 120253

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

3D-Net: Monocular 3D object recognition for traffic monitoring☆

Mahdi Rezaei [a,*], Mohsen Azarmi [a], Farzam Mohammad Pour Mir [b]

[a] University of Leeds, Institute for Transport Studies, United Kingdom
[b] Islamic Azad University, Science and Research Branch, Iran

ARTICLE INFO

Dataset link: https://codeocean.com/capsule/7713588/tree

Keywords:
Intelligent Transportation Systems (ITS)
Traffic monitoring
Autonomous vehicles
Computer vision
Object recognition
Machine learning
Camera auto-calibration
Future mobility

ABSTRACT

Machine Learning has played a major role in various applications including Autonomous Vehicles and Intelligent Transportation Systems. Utilizing a deep convolutional neural network, the article introduces a zero-calibration 3D Object recognition and tracking system for traffic monitoring. The model can accurately work on urban traffic cameras, regardless of their technical specification (i.e. resolution, lens, the field of view) and positioning (location, height, angle). For the first time, we introduce a novel satellite-ground inverse perspective mapping technique, which requires no camera calibrations and only needs the GPS position of the camera. This leads to an accurate environmental modeling solution that is capable of estimating road users' 3D bonding boxes, speed, and trajectory using a monocular camera. We have also contributed to a hierarchical activity/traffic modeling solution using short- and long-term Spatio-temporal video analysis to understand the heatmap of the traffic flow, bottlenecks, and high-risk zones. The experiments are conducted on four datasets: MIO-TCD, UA-DETRAC, GRAM-RTM, and Leeds-Dataset including various use cases and traffic scenarios.

1. Introduction

Smart video surveillance systems are becoming a common technology for various applications such as anomaly detection, security enhancement of crowded areas, sports events, as well as traffic monitoring (Min Gan, Fernando, & Molina-Solana, 2021), and congestion management in urban areas. Parallel to the technological improvements, the complexity of traffic scenes for automated traffic surveillance has also increased due to multiple factors such as urban developments, the mixture of classic and autonomous vehicles, population growth, and the increasing number of cyclists, pedestrians, and vulnerable road users (Nambiar, Shroff, & Handy, 2018). The rapidly growing number of surveillance cameras (over 20 million CCD cameras only in the USA and UK) in the arteries of cities, roads, and intersections, demonstrates the importance of video surveillance for city councils, authorities, and governments (Sheng, Yao, & Goel, 2021).

A large network of interconnected surveillance cameras can provide a special platform for further studies on traffic management and urban planning (Olatunji & Cheng, 2019). In contrast to traditional traffic monitoring systems, automated video surveillance has been researched for many years to improve the traffic flow, safety, and sustainability of transportation networks (Dutta et al., 2020). Computer Vision is one of the most investigated technologies for automated video surveillance. In automated traffic monitoring systems (ATMS) and Intelligent Transportation Systems (ITS), computer vision can extract a wide range of information from the traffic scenes (Poddar, Giridhar, Prabhu, Umadevi, et al., 2016).

Vehicle type recognition, vehicle counting, speed estimation, tracking, and trajectory estimation are examples of automated traffic scene analysis. Fig. 1 which is one of the sample outputs of this research, represents the road complexities including interactions between road users (pedestrians, vehicles, cyclists), their moving trajectories, speeds, and the density of the road users in various zones of the road. The top row of Fig. 1, shows a 3D road-user detection and classification, and the bottom row shows the real-time digital twin and replication of the same scene from the bird's eye view.

In such highly dynamic environments, the ability of real-time processing and accurate detection of simultaneous events becomes very crucial (Hu, Tan, Wang, & Maybank, 2004). Furthermore, an efficient traffic monitoring system should be capable of working with a grid of various interconnected cameras in different urban locations, where each camera may have a different resolution, different viewing angles, height, or focal length. The utilization of large a grid of cameras requires calibration of every single camera based on the intrinsic camera parameters and the mounting specification of each camera which is a costly and non-trivial task. As one of the main objectives of this

☆ This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101006664 as part of the Hi-Drive project.

* Corresponding author.
E-mail address: m.rezaei@leeds.ac.uk (M. Rezaei).

https://doi.org/10.1016/j.eswa.2023.120253
Received 6 February 2023; Received in revised form 21 April 2023; Accepted 22 April 2023
Available online 29 April 2023

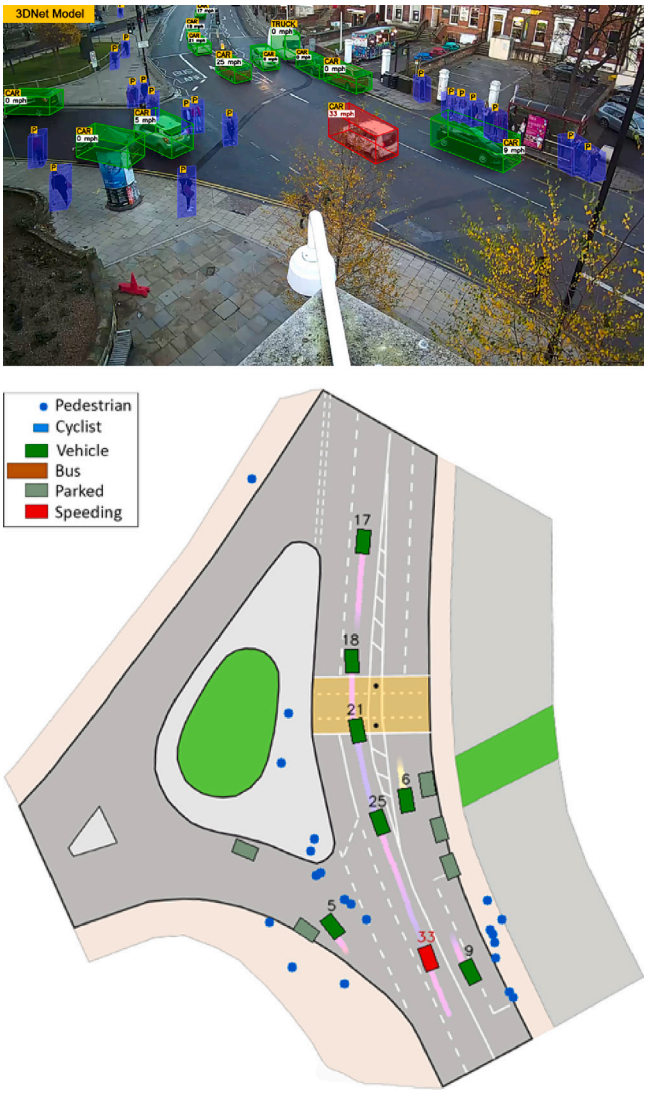


Fig. 1. 3D object detection and speed estimation (top row); Real-time digital twin and environment modeling of the same scene (bottom row).

research, we aim to cope with this challenge (i.e. developing an auto-calibration technique applicable to all cameras with any intrinsic and extrinsic specifications).

Although various methods of camera calibration such as vanishing-based techniques (Yang, Fang, & Tang, 2018) and multi-point calibrations techniques (Oliveira, Santos, & Sappa, 2015) have been introduced for bird's eye view mapping, fewer investigations have been conducted to introduce automated calibration methods.

On the other hand, vehicle and pedestrian classification is the heart of an ATMS, and this task should be handled by accurate object detection algorithms and tracking techniques (Brunetti, Buongiorno, Trotta, & Bevilacqua, 2018). In the past decade, the deployment of Deep Convolutional Neural Networks (DCNN) has led to significant advances in object detection. Not to mention the effectiveness and the accuracy of such models in complex, dynamic, noisy, and crowded traffic environments are not as good as object detection in controlled indoor environments.

Further challenges such as bad weather conditions, challenging lighting conditions (Rezaei, Terauchi, & Klette, 2015) during day and night, as well as occlusion, may also affect the performance of the object detection in traffic monitoring systems (Gawande, Hajari, & Golhar, 2020).

In this study, we contribute in four areas as follows:

- Adapting a custom DCNN for 3D road user recognition (vehicles, pedestrians, bikes) based on a single-stage and multi-head object detection architecture.
- Developing a multi-class object tracker to provide a continuous trajectory estimation among the occasional visual occlusions through the fused localization information.
- Developing a novel satellite/ground-based auto-calibration method for accurate localization and distance estimation of road users using surveillance cameras
- Automated short- and long-term traffic analysis to understand traffic bottlenecks, safety risks, and hazards for road users.

Detailed discussions will be provided in the next sections as follows: In Section 2 a comprehensive literature review is conducted on both classical and modern related works. Section 3 introduces our adapted methodology for object detection and tracking model for the Transport application followed by presenting a novel satellite/ground-based auto-calibration technique. Section 4 discuss the experiments, evaluations, traffic analysis, and comparisons with state-of-the-art. Finally, concluding remarks and possible routes to improve will be summarized in Section 5.

2. Related work

We review three types of related works to automated traffic surveillance systems (ATMS) including object detection methods, camera calibration approaches, and traffic monitoring solutions.

2.1. Road-user detection research

A series of classical studies have focused on detecting vehicles by providing motion-based solutions and background subtraction (BGS) techniques (Cheung & Kamath, 2004). Zhou, Gao, and Zhang (2007) have utilized support vector machine (SVM) classifier and dimensional reduction techniques to introduce an adaptive version of BGS which is robust in partial occlusion and bad illumination conditions. However, both method fails to detect stationary objects and they neglect to evaluate the crowded urban roads. Using Harris–Stephen corner detection algorithm was investigated to count the vehicle and estimates their speed in very basic traffic video scenes (Chintalacheruvu, Muthukumar, et al., 2012). In another approach, (Cheon, Lee, Yoon, & Park, 2012) have presented a traffic monitoring system in which they claim to find high-risk areas of accidents on the road throughout the day, whereas their histogram of oriented gradients (HOG) algorithm fails to detect the exact location and size of vehicles due to the shadow effect, and it has erroneous in the nights. Almost all of the classical methods were unable to distinguish between various categories of moving objects such as pedestrians, cars, buses, trucks, etc.

On the other hand, we have the emergence of deep learning solutions starting in the past decade. Popular CNN object detectors are mostly divided into two categories of single-stage (dense prediction) and two-stage (sparse prediction) detectors. The two-stage object detectors such as the RCNN family, consist of a region proposal stage and a classification stage (Jiao et al., 2019); while the single-stage object detectors such as Single-Shot Multi-Box Detector (SSD) (Liu et al., 2016), and the *You Only Look Once* (YOLO) sees the detection process as a regression problem, thus provides a single-unit localization and classification architecture (Jiao et al., 2019).

Arinaldi, Pradana, and Gurusinga (2018) reached a better vehicle detection performance using Faster-RCNN compared to a combination of the Mixture of Gaussian (MOG) and SVM models. Peppa et al. (2021), developed a statistical-based model, a random forest method, and a Long Short-Term Memory (LSTM) neural cell to predict the traffic volume for the upcoming minutes. Bui, Yi, and Cho (2020),

utilized YOLOv3 for automated vehicle detection by designing a multi-class distinguished-region tracking method to overcome the occlusion problem and lighting effects for traffic flow analysis.

Mandal, Mussah, Jin, and Adu-Gyamfi (2020) have proposed an anomaly detection system and compared the performance of different object detection including Faster-RCNN, Mask-RCNN, and YOLO. Among the evaluated models, YOLOv4 gained the highest detection accuracy. However, they have presented a pixel-based (pixel per second) vehicle velocity estimation that is not very accurate.

Arnold et al. (2019) have worked on 3D imaging of traffic scenes to distinguish the scene background from the foreground objects, and measure the objects' size, volume, and spatial dimensions. Zhang, Zheng, Xu, and Wang (2019) present a vehicle detector and tracking algorithm by clustering the LiDAR 3D point cloud data. Zhang, Xiao, Coifman, and Mills (2020) proposed a centroid-based tracking method and a refining module to track vehicles and improve speed estimations. Song, Yao, Ju, Jiang, and Du (2020) proposed a framework to detect roads, pedestrians, and vehicles using binocular cameras. In another multi-modal research, thermal sensor data is fused with the RGB camera sensor, leading to a noise-resistant technique for traffic monitoring (Alldieck, Bahnsen, & Moeslund, 2016).

Many studies including deep learning-based methods (Laga, 2019; Xie, Girshick, & Farhadi, 2016), and 3D object detection (Fernandes et al., 2021; Zhou, Fan, Cheng, Shen, & Shao, 2021) have tried to utilize multi-camera and sensors to compensate for the missing depth information in the monocular CCTV cameras, to estimate the position and speed of the object, as well as 3D bounding box representation from a 2D perspective images (Bhoi, 2019). However, the cost of applying such methods in large and crowded cities could be significant. As part of our research, we try to minimize such costs by using the existing infrastructure.

2.2. Camera calibration research

A camera transforms the 3D world scene into a 2D perspective image based on the camera's intrinsic and extrinsic parameters (Zhang, 2000) and camera calibration is a key stage in inverse perspective mapping, distance estimation, and vehicle speed estimation (Rezaei & Klette, 2017). However, most of the calibration studies have been conducted using checkerboards, straight-line vanishing points, and multi-point manual calibration procedures. Very limited studies have been conducted on automated camera calibration.

Dubská, Herout, Juránek, and Sochor (2015) extract vanishing points that are parallel and orthogonal to the road in a roadside surveillance camera image, using the moving trajectory of the detected cars and the Hough line transform algorithm. The algorithm suffers from the low accuracy of the Hough transform algorithm in challenging lighting and noisy conditions. In another study (Sochor, Juránek, & Herout, 2017), a Faster-RCNN model is employed to detect cars' edgelets and extract perpendicular vanishing points to estimate the camera parameters for calibration. Song et al. (2019) utilized an SSD object detector to detect cars and extract spatial features from the content of bounding boxes using optical flow to track them. They calculate two vanishing points using the moving trajectory of vehicles in order to automatically calibrate the camera. As a weakness, they consider a fixed average length, width, and height of cars to draw 3D bounding boxes.

However, all of the aforementioned calibration methods assume the road has zero curvature, therefore, the vanishing points are calculated based on straight-line roads. This will cause the model fails on curved roads.

In a different approach (Kim, 2009), consider more than six arbitrary corresponding points between the road-side camera image and another perpendicular view image of the same scene, followed by matching those points and estimating the calibration parameters using a revised version of RANSAC algorithm which can tolerate partial spatial noise, in point-wise feature matching. However, this requires the user manually sets these points, so, no automatic calibration procedure.

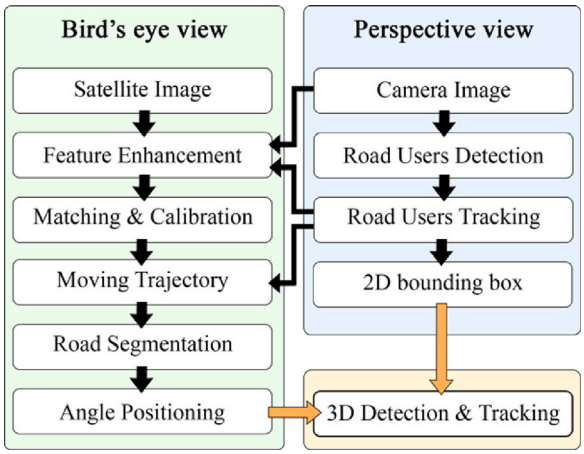


Fig. 2. The overall structure of hierarchical stages from 2D to 3D object detection.

2.3. Traffic modeling research

Some recent studies (Kashyap et al., 2022; Wang, Zhao, et al., 2022) have worked on traffic flow estimation and prediction using LSTM modules and Restricted Boltzmann Machines (RBM) to extract high-level spatial features from raw input progressively. Abdollahi, Khaleghi, and Yang (2020) suggest a learning-based approach to predict travel time. Chen et al. (2018) consider 3D CNNs, and Chen et al. (2020) uses graph-based CNNs (Cui, Henrickson, Ke, & Wang, 2019) and Graph Convolutional Recurrent Neural Network for dynamic traffic flow analysis. However, their analyses are based on the macro-scale traffic data in the cities and lack fine-grained assessments for real-time visual monitoring. On the other hand, mathematical traffic modeling research is designed to understand the important criteria for describing heterogeneous traffic flow based on the microscope urban and rural data (Mallikarjuna & Rao, 2011). The research emphasizes on that the size of vehicles and their lateral and longitudinal gaps for traffic analysis. However, the main argument here is that the mathematical modeling can be sometimes very abstract and cannot reflect the naturalistic and region-specific conditions of roads and environments (Wang, Tu, & Juang, 2021).

As part of our contribution, in the next section, we propose the development of an efficient digital twin of the environment by Spatio-temporal analysis of the scene, represented by a live visual heatmap of the environment. The proposed model aims at identifying highly congested areas, automated recognition of speeding violation zones, pedestrian favorite paths, and high potential crash zones for vehicles to pedestrians.

3. Methodology

The method will be presented in three hierarchical subsections. In Section 3.1 a re-structured and retrained YOLO-based detector as well as a multi-class object tracker (MCOT) model will be introduced for highly accurate road-user detection and tracking. This should not be confused with the general-purpose YOLO object detector's family. In Section 3.2, the novel auto-calibration technique (named SG-IPM) is detailed. In Section 3.3 the traffic modeling solution is proposed. These together lead to 3D detection and real-time digital twin of the traffic scene using a single CCTV camera. Fig. 2 summarizes the brief flowchart of the steps to be taken starting from a 2D camera image and satellite image feed to the 3D road-users detection, tracking, and traffic flow analysis.

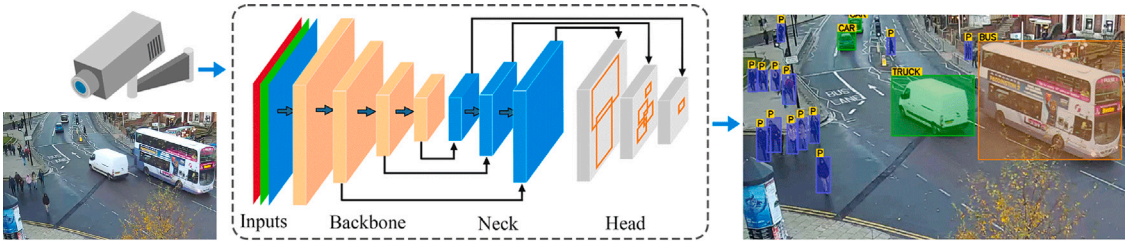


Fig. 3. Summarized structure of the dense (single-stage) deep convolutional neural network (DCNN) architecture for object detection applied to a roadside surveillance video.

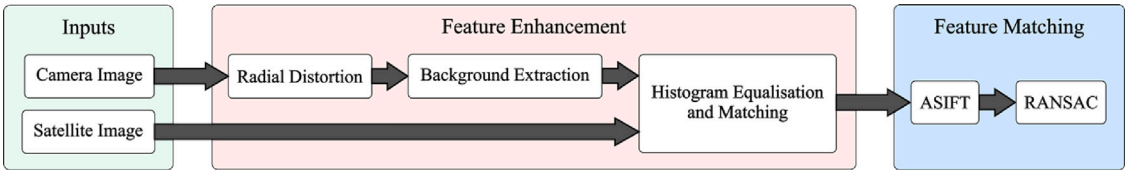


Fig. 4. The hierarchical structure of the feature matching model as the first stage of the proposed Auto-calibration model.

3.1. Road-users detection and tracking

Inspired by the latest YOLO family object detectors, trained on COCO (Lin et al., 2014) dataset, we developed a road user detection model called RUYOLO as per Fig. 3. The oranges blocks indicate *backbone* part in which the stacked convolutional layers is used to extract spatial feature from the input image. The blue blocks are *neck* slots of the model that is equipped with multiple partial connections, such as CSP (Wang et al., 2020) to alleviate vanishing gradient and boost feature propagation through the network. Moreover, a path aggregation method has been considered to enhance the semantic and spatial information. Conducting a set of comprehensive experimental studies on various loss functions, we optimized our model using Focal-Loss (Lin, Goyal, Girshick, He, & Dollár, 2017) and Distance-IoU loss (Zheng et al., 2020) in the *head* part of the network to enhance the classification results in the context of vehicles and pedestrians detection, during the training process.

We trained the RUYOLO based on the MIO-TCD (Luo et al., 2018) dataset that provides a variety of traffic video samples annotated by bounding boxes for 11 classes; Including pedestrian, articulated truck, bicycle, bus, car, motorcycle, motorized vehicles, non-motorized vehicles, pickup truck, single-unit truck, and work van to ensure a more accurate and customized model for our application.

The last part of the model produces a set D for each object of the image stream $(x_b, y_b, w_b, h_b, \mathfrak{s}, \mathbf{c})$, where x_b, y_b are center points, w_b, h_b are the width and height of the detected bounding boxes, $\mathfrak{s}$ is the objectness confidence score, and $\mathbf{c}$ is a vector of classification probabilities with a length equal to the number of classes. We consider the coordinates of the middle point at the bottom side of each bounding box as the reference point of the detected objects. This is the closest contact point of the vehicles and pedestrians to the ground (the road surface):

$$(\hat{x}, \hat{y}) = (x_b, y_b + \frac{h_b}{2}) \tag{1}$$

The SORT (Bewley, Ge, Ott, Ramos, & Upcroft, 2016) tracking algorithm has recently become popular for generic object tracking purposes. The algorithm assigns a unique ID to each object by computing the Intersection over Union (IoU) between detected bounding boxes in consequent frames of the input video. However, this process is only applicable to a single class tracking and each class needs to be dealt with separately. In some cases, the object detector assigns a new class to the same object which is not aligned with the object tracker estimation. As a common issue, in such cases, the SORT tracker sees the same object as a new object hence it assigns a new ID and loses the previous tracking.

To overcome this issue, we integrated a category vector $\acute{\mathbf{c}} \in \mathbb{W}^{1\times 11}$ to the baseline SORT tracker, followed by a Hungarian intra-frame association. This results in our multi-class object tracking (MCOT) solution without suffering from the false multi-ID assignment to the same object. The category vector is the one-hot encoded representation of the detected class vector $\mathbf{c}$, in which the highest class probability is set as 1 and the rest of the probabilities are suppressed to 0.

Exploiting the smoothing effect of the tracker would filter out the bouncing of detected categories through the sequence of frames. Also, it enables the tracker to calculate IoU between the bounding boxes of different categories. This yields a multi-object and multi-category ID assignment.

The state matrix of the tracker can be defined as follows:

$$\dot{\mathbf{x}} = [\ \hat{x}\ \ \hat{y}\ \ s_b\ \ r_b\ \ \dot{x}\ \ \dot{y}\ \ \dot{s}\ \ |\ \acute{\mathbf{c}}\ ]^T \tag{2}$$

where $s_b = w_b \times h_b$ denotes the object area, r_b is the aspect ratio, $\dot{x}$, $\dot{y}$ and $\dot{s}$ are the velocities of $\hat{x}$, $\hat{y}$ and s_b, respectively. Similarly, we represent the observation matrix of the revised tracker as follows:

$$\dot{\mathbf{z}} = [\ \hat{x}\ \ \hat{y}\ \ s_b\ \ r_b\ \ |\ \acute{\mathbf{c}}\ ]^T \tag{3}$$

In order to determine the trajectory of objects, we introduce two sets of V and P as the tracker-ID of detected vehicles and pedestrians, respectively.

The trajectory set of each vehicle (v_i) and pedestrian (p_i) can be calculated based on temporal image frames as follows:

$$\begin{aligned} M_{v_i} &= \{(\hat{x}^t_{v_i}, \hat{y}^t_{v_i}) \quad : \quad \forall t \in T_{v_i}\} \\ M_{p_i} &= \{(\hat{x}^t_{p_i}, \hat{y}^t_{p_i}) \quad : \quad \forall t \in T_{p_i}\} \end{aligned} \tag{4}$$

where T_{v_i} and T_{p_i} are the sets of frame-IDs of the vehicles v_i and pedestrians p_i and $(\hat{x}^t, \hat{y}^t)$ is the location of the object v_i or p_i at frame t.

Finally, the moving trajectories of all tracked objects are defined as the following sets:

$$\begin{aligned} M_V &= \{M_{v_i} \quad : \quad \forall v_i \in V\} \\ M_P &= \{M_{p_i} \quad : \quad \forall p_i \in P\} \end{aligned} \tag{5}$$

3.2. Camera auto-calibration

The intuition behind this part of the study is to develop an automatic inverse perspective mapping (IPM) camera calibration setup where and when no information about the camera's intrinsic and

mounting specifications is available. Despite the classical approaches of IPM which require multi-point camera calibration from the scene, we aim at making this study applicable to all CCTV traffic surveillance cameras.

Knowing only the Geo-location of a camera (GPS coordinates of the camera), we extract a top-view satellite image from the same location of the CCTV camera and aim at developing an auto-calibrated satellite-ground based inverse perspective mapping (SG-IPM) as follows. This is an end-to-end technique to estimate the planar transformation matrix $\mathbf{G}$ as per Eq. (27) in Appendix A. The matrix $\mathbf{G}$ is used to transform the camera perspective image to a bird's eye view image.

Let us assume (x, y) as a pixel in a digital image container $\mathbf{I} : \mathcal{U} \rightarrow [0, 255]^3$ were $\mathcal{U} = [[0; w-1] \times [0; h-1]]$ represents the range of pixel locations in a 3 channel image, and w, h are width and height of the image.

Using $\hat{()}$ to denote the perspective space (i.e. camera view), and $\grave{()}$ for inverse perspective space, we represent the surveillance camera image as $\hat{\mathbf{I}}$, the satellite image as $\grave{\mathbf{I}}$, and the bird's eye view image as $\check{\mathbf{I}}$ which is calculated by a linear transformation $\mathbf{G} : \hat{\mathbf{I}} \rightarrow \check{\mathbf{I}}$.

Since the coordinates of the bird's eye view image approximately match the satellite image coordinates (i.e. $\grave{\mathbf{I}} \approx \check{\mathbf{I}}$), the utilization of the transformation function $(\check{x}, \check{y}) = \Lambda((\hat{x}, \hat{y}), \mathbf{G})$ (as defined in Appendix A) would transform the pixel locations of $\hat{\mathbf{I}}$ to the $\grave{\mathbf{I}}$. Similarly, $\mathbf{G}^{-1}$ inverts the mapping process. In other words, $(\hat{x}, \hat{y}) = \Lambda((\check{x}, \check{y}), \mathbf{G}^{-1})$ transforms the pixel locations from $\check{\mathbf{I}}$ to $\hat{\mathbf{I}}$.

In order to solve the linear Eq. (27), we need at least four pairs of corresponding points in $\hat{\mathbf{I}}$ (ground-based image) and $\grave{\mathbf{I}}$ (satellite-based image). The feature points should be robust and invariant to rotation, translation, scale, tilt, and also partial occlusion in case of high affine variations.

Fig. 4 represents the general flowchart of the proposed SG-IPM technique, which is fully explained in the following two sub-sections-*feature enhancement* and *feature matching*:

3.2.1. Feature enhancement

In order to increase the spatial resemblance between $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ images for finding strong similar points, we applied a radial distortion correction technique on the camera image to remove the barrel-shape radial noise (Dubská et al., 2015) due to the camera lens and optics imperfection.

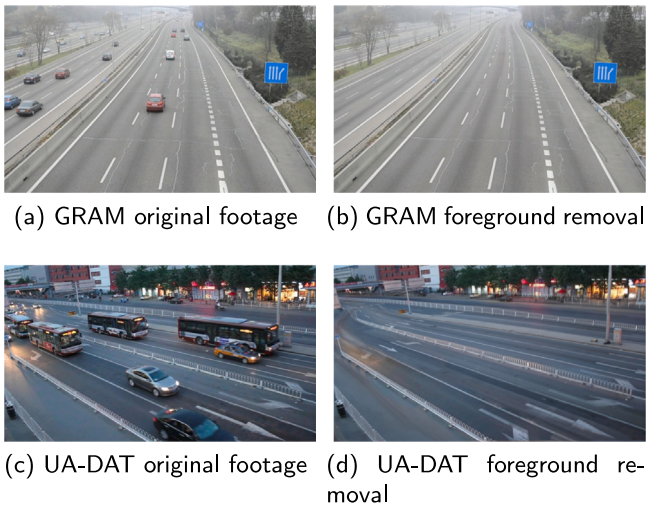
To separate the background object from the foreground objects and road users in the traffic scene $\hat{\mathbf{I}}$, an accumulative weighted sum over the intensity value was applied for a period of n_t (frames) to remove the effect of the temporal pixel value changes:

$$\hat{\mathbf{B}}^t = (1-\alpha)\hat{\mathbf{B}}^{t-1} + (\alpha\ \hat{\mathbf{I}}^t) \quad , \quad 1 \le t \le \mathsf{n}_t \tag{6}$$

where $\hat{\mathbf{B}}$ is an accumulative variable and $\hat{\mathbf{B}}^0$ will be equal to the initial input frame ($\hat{\mathbf{I}}^0$). The weighted coefficient α determines the importance of the next incoming frame. Our experiment shows that $\alpha = 0.01$, and $\mathsf{n}_t \approx 70$ frames are usually sufficient to remove the foreground objects in most mid to high-speed roads with a moderate traffic flow. Fig. 5 shows sample outputs of the automated foreground removal of the background extraction after 70 frames, applied to two different roads and traffic scenarios. A color correlation-based histogram matching is also applied to transfer the hue and luminance of $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ into the same range (Niu, Lu, & Wang, 2018). This helps to decrease the color variation of both images.

3.2.2. Feature matching

Inspired by the Affine Scale Invariant Feature Transform (ASIFT) (Yu & Morel, 2011), we apply random affine transform to create different viewpoints of both $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$. This allows us to have more sample images with different latitude and longitude angles compared to the camera and satellite images. Consequently, this will increase the chance of matching similar feature pairs between $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$. However, there might be some outliers between the matching features causing



(a) GRAM original footage (b) GRAM foreground removal

(c) UA-DAT original footage (d) UA-DAT foreground removal

Fig. 5. GRAM-RTM dataset (Guerrero-Gomez-Olmedo, Lopez-Sastre, Maldonado-Bascon, & Fernandez-Caballero, 2013), and UA-DAT dataset (Wen et al., 2020).

inaccurate estimation of the matrix $\mathbf{G}$. To remove these outliers, a Random Sample Consensus (RANSAC) is used, as an iterative learning algorithm for parameter estimation (Fischler & Bolles, 1981). In each iteration, the algorithm randomly samples four corresponding pairs among all matching points between $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$. Then, it calculates the $\mathbf{G}$ matrix using the collected samples and performs a voting process on all matching feature pairs in order to find the best matching samples.

Considering $\hat{l}_f$ and $\grave{l}_f$ as the locations of the matching pairs, the following criterion is defined to evaluate the best candidate pairs:

$$F_n = \begin{cases} 1 & d(\Lambda(\hat{l}_f, \mathbf{G}),\ \ \grave{l}_f) < \tau_{\mathbf{Z}} \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

where F_n is the result of voting for the nth pair, $\tau_{\mathbf{Z}}$ is a distance threshold to determine whether a pair is an inlier, and d is the Euclidean distance measure. Consequently, the total number of inlier votes ($\hbar_i$) for the matrix $\mathbf{G}$ in the ith iteration will be calculated as follows:

$$\hbar_i = \sum_{n=1}^{\eta} F_n \quad , \quad i \in \zeta \tag{8}$$

where η is the total number of matching feature pairs, and ζ is the total number of iterations which is defined as follows:

$$\zeta = \frac{\log(1-\rho)}{\log(1-\epsilon^{\gamma})} \tag{9}$$

where ϵ is the probability of a pair being inlier (total number of inliers divided by η), γ is the minimum number of random samples (4 feature-pairs in our setting, which is the least requirement in order to calculate the matrix $\mathbf{G}$), and ρ is the probability of all ς sampled pairs being inliers in an iteration.

After the end of the iterations, the matrix $\mathbf{G}$ with the highest vote will be elected as the suitable transformation matrix between $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$. Fig. 6 represents a sample feature and inlier matching process applied to a real-world scenario.

Finally, the matrix $\mathbf{G}$ is used in the Λ function to map the object's coordinates from $\check{\mathbf{I}}$ to $\hat{\mathbf{I}}$ coordinates (as defined in Appendix A Eq. (28)).

$$(\check{x}, \check{y}) = \Lambda((\hat{x}, \hat{y}), \mathbf{G}) \tag{10}$$

For better visualization and comparison, Fig. 7 represents a very accurate overlapping and the intersection of the road, after mapping of the estimated matrix $\mathbf{G}$ on $\hat{\mathbf{I}}$ coordinates.

(a) Feature matching results

(b) RANSAC-based inlier matching

Fig. 6. Feature matching applied on a sample road scene in Leeds via a CCTV surveillance camera (right) and the corresponding satellite view of the same location (left side).

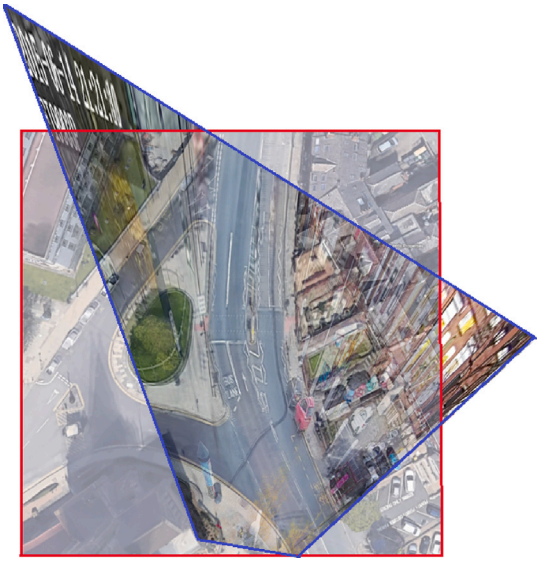


Fig. 7. Overlapping the estimated BEV image $\check{\mathbf{I}}$ (blue) to the ground truth satellite image $\grave{\mathbf{I}}$ (red) of the same location.

3.3. Environment modeling and traffic analysis

Automated analysis of traffic scene videos via surveillance cameras is a complex task. This is mainly due to the existence of various types of objects such as trees, buildings, road users, banners, etc in various sizes and distances. Occlusion and lighting conditions are additional parameters that make it non-trivial. In this section, we elaborate on our techniques for providing an abstract visual representation of the environment, objects of interest, traffic density, and traffic flow. In order to achieve a 3D modeling and representation of the road users (e.g. vehicles), we require to identify and recognize the following properties for the road users and the road scene:

- Vehicle's speed (ϑ)
- Vehicle's heading angle (θ)
- Road boundaries

Before proceeding any further, we need to ensure an accurate mapping from the image plane to the bird's eye view image. We have observed bird's eye view projected points normally suffer from some noise caused due transformation ($\mathbf{G}$). To alleviate this noise and recover the natural movement of objects (especially the vehicle's dynamics, and moving trajectory, we applied a constant velocity Kalman filter (as defined in Appendix B). This significantly improves the tracking performance to provide a more accurate speed and heading angle estimation.

3.3.1. Speed estimation

Assuming the location of vehicle v_i in time t as $\check{l}^t_{v_i} = (\check{x}^t_{v_i}, \check{y}^t_{v_i})$ and $t-1$ as $\check{l}^{t-1}_{v_i} = (\check{x}^{t-1}_{v_i}, \check{y}^{t-1}_{v_i})$ in the trajectory set M_{v_i}, the velocity of v_i can be calculated as follows:

$$\vartheta_{v_i} = \frac{d(\check{l}^t_{v_i}, \check{l}^{t-1}_{v_i})}{\Delta t} \times \iota \tag{11}$$

where Δt is the time difference in seconds, and ι is the length of one pixel in meters (pixel-to-meter ratio).

To calculate ι, we consider a common ground truth measure, a standard object, sign, or road marking with a known size in the scene, such as the width of the 2-lane city roads (e.g. 7 m standard in the UK) or the length of the white lane markings (e.g. 3 m standard in Japan) as a real-world distance reference. Dividing the real-distance reference by the number of pixels in the same region of the satellite image gives us the pixel-to-meter ratio (ι).

3.3.2. Heading angle estimation

We calculate the heading angle of a vehicle as follows:

$$\theta_{v_i} = \theta(\check{l}^t_{v_i}, \check{l}^{t-1}_{v_i}) = \tan^{-1}\left(\frac{\check{y}^t_{v_i} - \check{y}^{t-1}_{v_i}}{\check{x}^t_{v_i} - \check{x}^{t-1}_{v_i}}\right) \tag{12}$$

The angle estimation is very sensitive to the displacement of vehicle bonding box noise, and even a small noise in localization can lead to a significant change in the heading angle. Knowing that in the real world, the heading angle of vehicles cannot change significantly in a short period of time (e.g. between two consequent frames), we introduce an efficient Angle Bounce Filtering (ABF) method to restrict sudden erroneous angle changes between the current and previous angle of the vehicle:

$$\Delta\theta_{v_i} = \theta^t_{v_i} - \theta^{t-1}_{v_i} \tag{13}$$

where $\Delta\theta_{v_i}$ is in the range of $[-180°, 180°]$. In order to suppress high rates of the changes, we consider a cosine weight coefficient (w) as follows:

$$\mathrm{w} = \frac{\cos((4\pi \times \tilde{\Delta}) + 1)}{2} \tag{14}$$

where $\tilde{\Delta}$ is the normalized value of $\Delta\theta_{v_i}$ within the range of $[0, 1]$. The coefficient yields to "0" when the $\Delta\theta_{v_i}$ approaches to $\pm 90°$ to neutralize the sudden angle changes of the vehicle. Similarly, the coefficient yields to "1" when the $\Delta\theta_{v_i}$ approaches to $0°$ or $\pm 180°$ to maintain the natural forward and backward movement of the vehicle. Fig. 8 illustrates the smoothed values of w by green color spectrum. The darker green, the lower the coefficient.

Finally, we rectify the vehicle angle as follows:

$$\tilde{\theta}^t_{v_i} = \theta^{t-1}_{v_i} + (\mathrm{w} \times \Delta\theta_{v_i}) \tag{15}$$

In some cases the moving trajectory may not be available; for instance, some vehicles are stationary (or parked) during their entire presence on the scene. In such cases, the angle of the nearest road border is considered the reference angle for the stationary vehicle. This idea is shown in Fig. 9. Utilizing a seeded region growing (SRG)

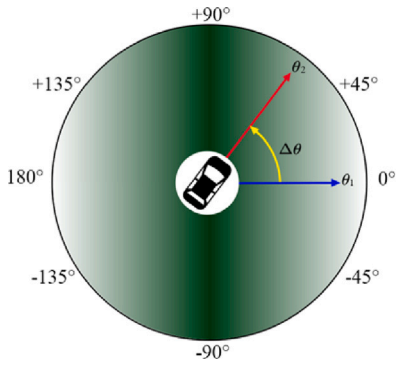


Fig. 8. $\Delta\theta$ cosine suppression operation. The darker zones receive lower coefficients which in turn suppress any large and sudden angular changes between two consequent frames.

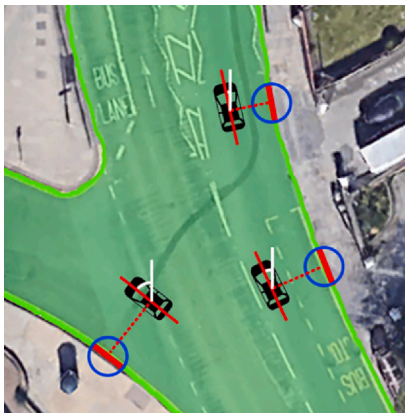


Fig. 9. Reference angle estimation with respect to the nearest road boundary. The detected boundaries are shown with light green lines.

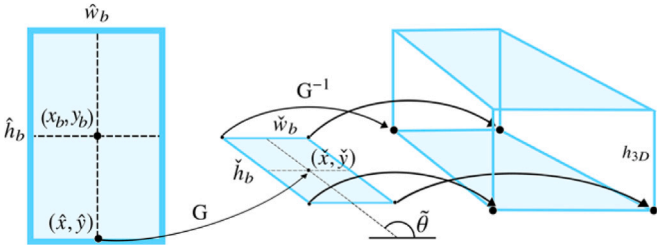
method (Ahmad, 2021) we segment the road regions in $\check{\mathbf{I}}$ domain. Then a morphological dilation operation is applied to expand the segmented area and fill the small gap regions, followed by an erosion operation to smooth the road region by removing the sharp edges and spikes of the road boundaries. Fig. 9 green regions represent sample segmentation results.

Fig. 10(a), 10(b), 10(c) show the hierarchical steps of our approach from 2D to 3D conversion on a sample surveillance camera footage from Leeds, England.

3.3.3. 2D to 3D bounding box conversion

In order to determine the occupied space by each road user in $\hat{\mathbf{I}}$ the scene and in each lane, we convert a 2D bounding box (Fig. 10(a)) to a cubical 3D bounding box by estimating 8 cube's corners. The cube's floor consists of 4 corner points and corresponds to a rectangle in the $\check{\mathbf{I}}$ domain (Fig. 10(b) the middle shape). This rectangle indicates the area of the ground plate which is occupied by the object and can be addressed with the center $(\check{x}, \check{y})$, the height $\check{h}_b$ and the width $\check{w}_b$. The $\check{h}_b$ and $\check{w}_b$ are determined based on ground truth knowledge about the approximate 2D dimensions of the corresponding object's class in the real world. Our 2D object detector model identifies 10 classes of vehicles, including bicycles, motorcycles, articulated trucks, pickup trucks, single-unit trucks, vans, minibusses, double-decker buses, mini cars, and standard sedan cars. Having the center point of the bounding box and the class of the object, the width and length of the object will be extracted from the object's class ground truth (shown as w_b and h_b) in Fig. 10(b). (e.g. 2.55 m×4.95 m meter for Double-Decker buses in the UK).

(a) Detected objects in 2D bounding boxes



(b) 2D to 3D bounding box conversion

(c) Final 3D representation

Fig. 10. 2D to 3D bounding box conversion process in four categories of vehicle/truck, pedestrian, bus, and cyclist.

Observing the video data it can be seen that regardless of the class of object, the height of the objects always fits within the 2D bounding box. So, to calculate the height of the 3D object, we fit the 3D bounding box, tangent to the upper edge of the 2D bounding box.

In order to apply the real size of vehicles in a pixel unit, we divide their real-world size by the pixel-to-meter ratio (ι), as explained in Section 3.3.1, Speed Estimation.

For each vehicle, the rectangle is rotated and aligned with the estimated heading angle $\tilde{\theta}_{v_i}$ to represent the object's movement direction. Then, the four corners of the resulting rectangle are converted to $\hat{\mathbf{I}}$ domain using the $\mathbf{G}^{-1}$ matrix and considered as the corners of the cube's floor. Afterward, we add h_{3D} to the y axis of the floor corners to indicate the 4 points of the cube's roof (Fig. 10(c)).

The height of the cube for all road users, except the pedestrians, is set $h_{3D} = \beta \times \hat{h}_b$, where $\beta = 0.6$ is determined by our experiments as a suitable height coefficient for the detected bounding boxes in the $\hat{\mathbf{I}}$ domain. The cube's height for pedestrians is equal to the height of the detected bounding box in the perspective domain ($h_{3D} = \hat{h}_b$).

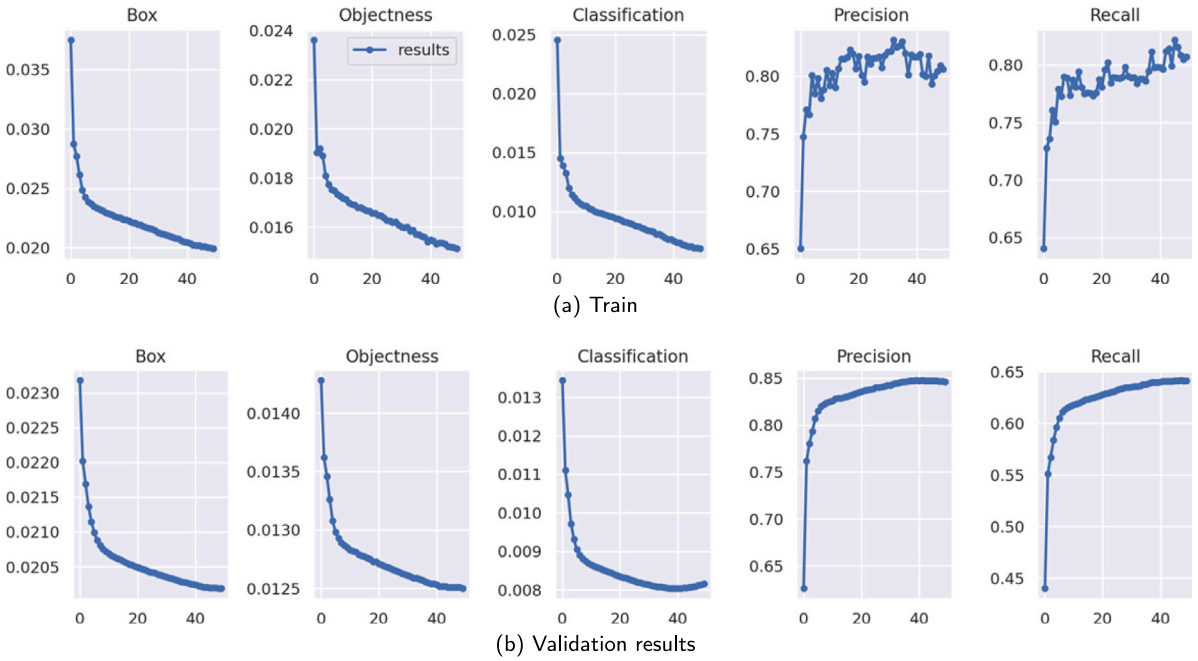


Fig. 11. Error minimization graphs of the model in training and validations phases, after 50 epochs.

4. Experiments

In this section, three sets of experiments will be conducted: The performance of our camera calibration method, the proposed 3D road user detector, and the effectiveness of the traffic monitoring model.

Calibration Evaluation

The parameters including camera intrinsic, distortion coefficients, and camera extrinsic are evaluated by comparing the estimated **K**, **R**, and **T** matrices (as defined in Appendix A) with the actual estimated values of the camera using chessboard calibration (Zhang, 2000). Table 1 shows the uncertainty of each estimated parameter using Standard Error SE metric. The standard error can be used to calculate confidence intervals.

We examine the calibration at different times in the Leeds video, where the ambient light and traffic density seem to be decent to apply feature enhancement (Section 3.2.1) and matching (Section 3.2.2). The results indicate the auto-calibration technique could approximate extrinsic parameters better than the intrinsic matrix. This might be due to the satellite image that shares more information for rotation and translation values; however, on the other hand, the intrinsic parameters (focal length, principal point, and lens redial distortion) cannot be easily estimated or assessed.

Furthermore, given we know the ground truth measurements for the road width of the scene (in front of the Parkinson building, Leeds, UK), we also compared the road width of the auto-calibrated image against the ground truth measures at 30 different points of the scene. This shows a mean displacement error (map shift error) $\mu_d = 0.51$ m with an $\sigma_d = 1.1$ and also an average road width size error of $\mu_s = 0.34$ m and an $\sigma_s = 0.7$. This is a pretty good result for a two-lane road with a ground truth width of ≈ 7.0 meter in our dataset.

4.1. Performance evaluation

We elaborate on our DNN-based model in terms of training data, learning and hyperparameters, and model evaluation on testing data in this sub-section.

Table 1
Confidence interval of estimated camera parameters using $SE = \frac{\sigma}{\sqrt{n}}$, during different time intervals of the Leeds Dataset.

Time (': '')	K	R	T
00:05–00:15	0.71	0.84	0.82
02:20–02:30	0.70	0.81	0.81
19:28–19:44	0.72	0.88	0.84
22:10–23:17	0.61	0.79	0.77
45:33–45:40	0.69	0.79	0.75
50:55–51:10	0.67	0.72	0.79

4.1.1. Datasets

MIO-TCD dataset (Luo et al., 2018) was considered to train the proposed RUYOLO model, containing 648,959 images and 11 traffic-related annotated categories. The dataset includes cars, pedestrians, bicycles, buses, three types of trucks, two types of vans, motorized vehicles, and non-motorized vehicles. The dataset is collected at different times of the day and different seasons of the year by thousands of traffic cameras deployed all over Canada and the United States.

As per Table 2, we also considered two more traffic monitoring datasets of UA-DETRAC (Wen et al., 2020) and GRAM Road-Traffic Monitoring (GRAM-RTM) (Guerrero-Gomez-Olmedo et al., 2013) to test the models under various weather and day/night lighting conditions.

Moreover, we set up a surveillance camera at one of the highly interactive intersections of Leeds City (at the 3-way intersection of Woodhouse Lane, Blenheim Terrace, and Cavendish Road) to further evaluate the performance of the model on real-world scenarios. The dataset is called the "Leeds dataset" and consists of 940,000 video frames from the live traffic.

4.1.2. Training

Exploiting the pre-trained weights of the 80-class COCO dataset as initial weights of the model, we re-trained the RUYOLO with different sizes and numbers of head architectures on the MIO-TCD dataset.

Four sizes of learning parameters were considered: The "small" with 7.5 million parameters as a lightweight version, the "medium" version (21.8 million), the "large" (47.8 million), and the "xlarg" version with 89 million learnable parameters. We also defined two types of head

Table 2
Specifications of the test datasets used in this research, including various weather conditions, resolutions, frame rates, and video lengths.

Name	Weather	Length (frame)	Resolution	*fps*
UA-DETRAC (2020)	Sunny, Rainy, Cloudy, Night	140000	960 × 540	25
GRAM-RTM (2013)	Sunny, Foggy	40345	1200 × 720	30
Leeds Dataset (2023)	Day, Sunset, Night	940000	1920 × 1080	30

modules, (three and four head outputs) to classify different sizes of objects.

Each head gives a particular scale of features from the neck and takes bounding box and class prediction. Division scales for the '3-head' module are $[1, 8, 16]$, and $[1, 8, 16, 32]$ for the '4-head' module. In the training phase (Fig. 11(a)), we minimized the loss function of the RUYOLO, based on a sum of three loss terms including the "C-IoU loss" as the bounding box regression loss, "objectness confidence loss", and "binary cross entropy" as the classification loss.

In order to choose the optimal learning rate and avoid long training time, we used a one-cycle learning rate (Smith, 2018). This gradually increases the learning rate to a certain value (called the warm-up phase) followed by a decreasing trend to find the minimum loss, while avoiding local minima. In our experiments, we found the minimum and maximum learning rates of 0.01 and 0.2 as the optimum values.

The SOTA methods mentioned in Table 3 indicated by * sign, are retrained under the same experimental conditions, hardware, and software platform as for our proposed model, RUYOLO. We used 60 epochs with a 0.01 learning rate in all experiments and stopped the training process when it reached a stable convergence with no further improvements.

Fig. 11 illustrates the analytic graphs of the training and validation processes. As per the classification graphs (Fig. 11(a)), the training loss starts decreasing around epoch 35, while the validation loss starts increasing (Fig. 11(b)). This is a sign of over-fitting in which the model starts memorizing the dataset instead of learning generalized features. To avoid the effects of over-fitting, we choose the optimal weights which yield the minimum validation loss.

4.1.3. Evaluation of road users detection

Table 3 compares the performance of the proposed model with 16 other state-of-the-art object detection methods on the challenging dataset of MIO-TCD. Two metrics of *mAP* and speed (*fps*) are investigated. As can be seen, the RUYOLO model has achieved a considerable increase in mean average precision compared to the latest member of the YOLO family the YOLOv7 (Wang, Bochkovskiy, & Liao, 2022) (84.6% versus 83.1%).

The experiments also proved that the 3-head version of the model provides more efficiency in traffic monitoring than the 4-head version. This can be justified because the fourth head scales the features maps by 16 strides to detect large objects, whereas the traffic video footage normally does not have large objects due to the relatively far distance of the installed cameras from the road objects.

The RUYOLO xLarge and Large, with 3 heads reach the highest accuracy of 84.6% on the MIO-TCD benchmark dataset. Although the xLarge model has more parameters to learn features, the network complexity is greater than what is required to learn the features in the dataset. This prevents the accuracy to go beyond 84.6%. Also, it suffers from the lack of adequate speed to perform in real-time performance. The lightweight version of the model reaches the highest rate of speed (123 fps), while the model has sacrificed the accuracy by −1.8% in comparison to the highest rate of 84.6%. Whereas the Large, 3-head has the same mAP score and provides a real-time performance of 36.5 fps. This makes the model more suitable for cases in which heavy post-processing procedures are involved.

Table 4 shows the test results of the detection model and the multi-class object tracking (MCOT) algorithm in an ablation study on GRAM-RTM and UA-DETRAC datasets. The GRAM-RTM dataset provides three video types including high (1200 × 720), moderate (800 × 480), and low (600 × 360) resolution to challenge the detection algorithm.

4.1.4. Evaluation of road users tracking

Exploiting the MCOT algorithm leads to a significant increase in precision by +1.8% and +2.4% for both GRAM-RTM and UA-DETRAC, respectively (Table 4). During the detailed review of our experiments, we noticed that the detection algorithm frequently fails to detect highly occluded objects. However, our tracking algorithm could cover those circumstances and helps to maintain sustainable precision. Equipping the detector to our MCOT tracker, the algorithm recall rates surged remarkably by +3.8% and +2.9% on UA-DETRAC and GRAM-RTM datasets, respectively. This however also adds a very minor overload (approximately −0.5 *fps*) on the model run-time.

To evaluate the MCOT algorithm, we used the GRAM-RTM benchmark which includes four vehicle types (car, truck, bus, motorbike) plus their trajectory and poses information Table 5 shows the Mean Square Errors (MSE) of the model for trajectory and head angle estimation of the MCOT model compared to the given benchmark.

4.1.5. Evaluation of 3D bounding boxes

Evaluating the accuracy of 3D object detectors against LiDAR data points is one of the possible approaches to validate the effectiveness of 3D detections. However, to the best of our knowledge, there is no publicly available dataset for 3D traffic monitoring using Camera-Lidar. Given that we know the ground truth width of the road lanes and ground truth dimensions of some of the objects in the scene (such as Leeds City double-decker buses, Toyota Yaris, Ford Transit, etc.), we randomly selected 32 instances of 3D objects from 5 classes of vehicles (as per Table 6) in the scene and compared the predicted bounding boxes against the ground truth value as well as the mean absolute percentage error (MAPE) of the sizes.

Regarding the height coefficient (β), we selected 50 sample data-pair points $(\hat{h}_b, h_{3D})$ from manually annotated ground truth 2D and 3D bounding boxes for frequent classes of vehicles, followed by a linear regression line fitting through the data-pair points. The beta value of 0.603 is equal to the gradient of the regression line.

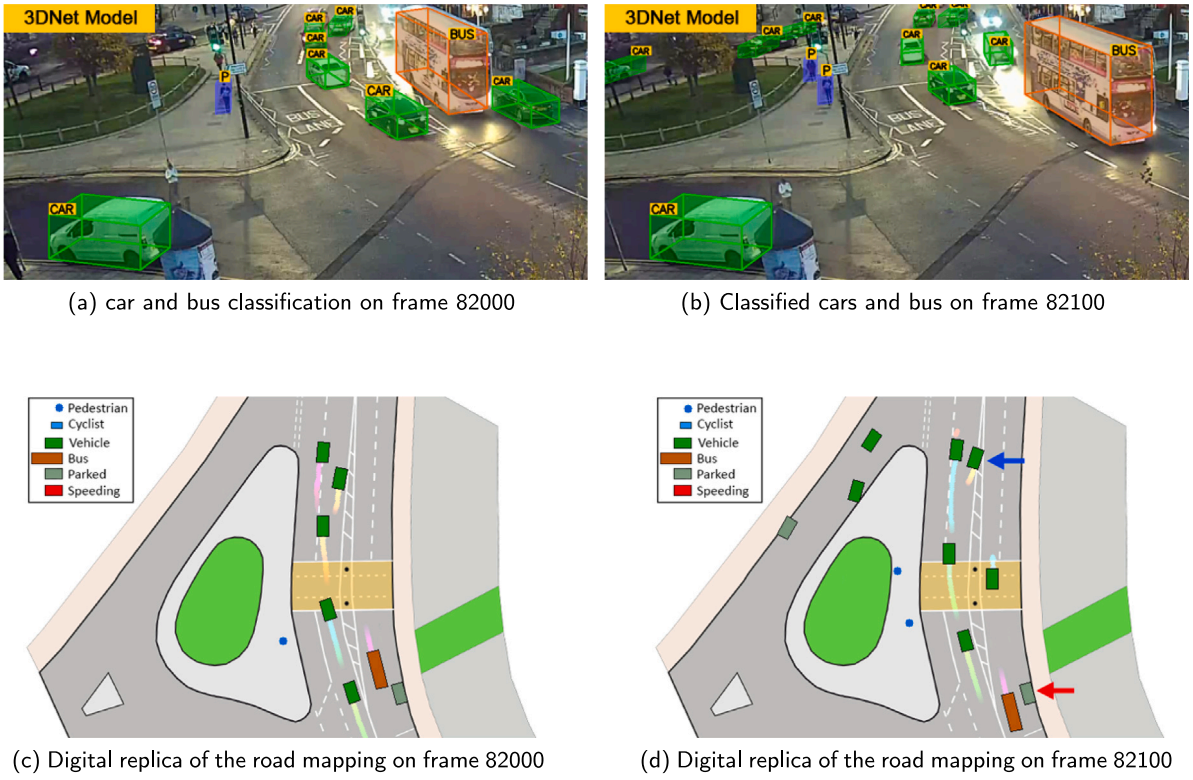
Fig. 12, top row, shows the road users' detection and 3D localization results. Fig. 12, bottom row, shows the traffic scene modeling as a digital twin of the traffic flow. Such live information would be very useful for city councils, authorities, and policymakers, and as an extra source of processed data for connected automated vehicles (CAVs) traversing around the same zone, particularly in dealing with corner cases and complicated traffic scenarios.

In Fig. 12 we are also trying to show the efficiency of the heading angle estimation and the tracking system in case of full occlusions. As can be seen, one of the cars in Fig. 12(d) (shown by the blue arrow) is taking a U-turn and the heading angle of the car is identified at frame 82100. This is done by comparing the previous position at frame 82000. Knowing the position and the heading angle of the vehicle at frames 82000 and 82100, the 3D bounding box of the vehicle is also determined.

In another complicated case in the same scene, one of the cars has been entirely occluded by a passing bus at frame 82100 (indicated with a red arrow). However, the car is fully traced by utilization of the spatio-temporal information and tracking model at frame 82000 and beyond.

Table 3
A comparison of mean average precision (mAP) rate between the developed models and 19 other SOTA models on the MIO-TCD dataset. The accuracy scores of three truck categories including Articulate Truck, Pickup Truck, and Single Unit Truck are averaged and presented in the column- "Trucks".

	Method	*fps*	mAP	Bicycle	Bus	Car	Motorcycle	Motorized vehicle	Non-motorized vehicle	Pedestrian	Work van	Trucks
SOTA	Context Model (2017)	–	77.2%	79.9%	96.8%	93.8%	83.6%	56.4%	58.2%	42.6%	79.6%	86.1%
	RFCN-ResNet-Ensemble (2017)	–	79.2%	87.3%	97.5%	89.7%	88.2%	**62.3%**	59.1%	48.6%	79.9%	86.4%
	Faster-RCNN (2018)	9	70.0%	78.3%	95.2%	82.6%	81.1%	52.8%	37.4%	31.3%	73.6%	79.2%
	SSD-512 (2018)	16	77.3%	78.6%	96.8%	94.0%	82.3%	56.8%	58.8%	43.6%	80.4%	86.4%
	SSD-300 (2018)	16	74.0%	78.3%	95.7%	91.5%	78.9%	51.4%	55.2%	37.3%	75.0%	83.5%
	YOLOv1 (2018)	19	62.7%	70.0%	91.6%	77.2%	71.4%	44.4%	20.7%	18.1%	69.3%	75.5%
	YOLOv2-MIOTCD (2018)	18	71.8%	78.6%	95.1%	81.4%	81.4%	51.7%	56.6%	25.0%	76.4%	81.3%
	YOLOv2-PascalVOC (2018)	18	71.5%	78.4%	95.2%	80.5%	80.9%	52.0%	56.5%	25.7%	75.7%	80.4%
	EfficientDet (Midiapite)* (2019)	25	79.6%	88.6%	95.5%	92.0%	91.5%	57.9%	64.8%	62.3%	80.0%	84.3%
	Detectron2* (2019)	11	72.3%	79.2%	95.3%	82.9%	82.2%	53.6%	39.4%	37.3%	76.1%	79.8%
	Adaptive Ensemble (2020)	–	74.2%	82.2%	95.7%	91.8%	87.3%	60.7%	45.7%	47.9%	63.8%	80.5%
	YOLOv4 * (2020)	24	80.4%	89.2%	95.8%	91.6%	91.5%	58.6%	63.9%	63.4%	79.0%	83.7%
	YOLOR * (2021)	26.7	77.9%	88.1%	95.8%	90.7%	90.2%	44.6%	64.8%	63.5%	80.2%	83.6%
	YOLOX * (2021)	27.1	79.6%	90.7%	95.9%	92.6%	91.9%	46.4%	65.2%	68.1%	81.7%	84.4%
	Stream-YOLO (2022)	37.4	81.3%	91.6%	96.1%	93.2%	93.5%	50.3%	67.9%	69.9%	81.8%	87.7%
	PP-YOLOv2-ResNet101 (2021)	48.9	80.4%	90.0%	95.8%	93.0%	92.2%	47.1%	68.1%	71.3%	80.7%	85.6%
	PP-YOLOE-m * (2022)	59.3	80.9%	91.2%	96.1%	94.1%	92.7%	47.9%	68.6%	71.4%	82.0%	83.9%
	PP-YOLOE-l * (2022)	35.7	82.4%	92.6%	96.5%	95.3%	93.5%	48.6%	69.7%	75.9%	83.3%	86.1%
	YOLOv7*(2022)	45.1	83.1%	91.9%	98.5%	94.8%	92.7%	50.9%	70.7%	75.8%	83.2%	88.7%
RUYOLO (Ours)	Small, 3-head	**123.5**	82.8%	91.6%	98.3%	95.5%	94.1%	50.5%	65.6%	70.1%	81.8%	87.8%
	Medium, 3-head	60.60	84.1%	92.4%	98.4%	95.9%	**94.3%**	51.7%	68.8%	74.8%	83.3%	88.6%
	Large, 3-head	**36.50**	**84.6%**	92.5%	**98.7%**	**96.0%**	**94.3%**	51.7%	**70.1%**	**77.4%**	**83.8%**	**88.8%**
	xLarge, 3-head	20.16	**84.6%**	**92.7%**	**98.7%**	**96.0%**	94.1%	51.7%	**71.2%**	**76.2%**	**83.8%**	**88.8%**
	Small, 4-head	117.6	80.9%	91.2%	97.8%	95.1%	91.7%	48.4%	61.9%	64.3%	80.0%	86.6%
	Medium, 4-head	54.90	82.9%	92.2%	98.4%	95.5%	93.1%	50.0%	66.8%	69.5%	82.1%	88.1%
	Large, 4-head	33.00	83.4%	**92.9%**	98.4%	95.7%	93.7%	50.6%	68.0%	71.3%	82.6%	88.0%
	xLarge, 4-head	19.20	83.7%	91.8%	98.4%	95.7%	93.5%	50.8%	69.0%	72.2%	83.4%	88.5%



(a) car and bus classification on frame 82000 (b) Classified cars and bus on frame 82100

(c) Digital replica of the road mapping on frame 82000 (d) Digital replica of the road mapping on frame 82100

Fig. 12. The outputs of adapted RUYOLO-large 3-head and multi-class object tracking algorithm for road-user detection and environment modeling.

4.2. Environment modeling and traffic analysis

In order to provide smart traffic monitoring analysis, we define five possible states for vehicles and pedestrians in the scene as follows:

- **Normal Vehicles**: a set $\mathcal{V}$ contains all vehicles in $\check{\mathbf{I}}$ domain traveling within the defined speed limit (e.g. max: 30 mph)
- **Normal Pedestrians**: a set $\mathcal{H}$ contains all pedestrians in $\check{\mathbf{I}}$ domain who are walking in pedestrian pathways and are not in the high-risk distance (i.e. less than 1 meter) to any moving vehicles.

Table 4
Detection performance of our model on two auxiliary traffic-related datasets.

Dataset	MCOT	Precision	Recall	Speed (fps)
UA-DETRAC	✗	98.0%	95.9%	36.95
UA-DETRAC	✓	99.8%	99.7%	36.45
GRAM-RTM	✗	97.3%	96.6%	36.91
GRAM-RTM	✓	99.7%	99.5%	36.39

Table 5
Performance evaluation of MCOT using GRAM-RTM benchmark.

Video	Trajectory MSE	Heading angle MSE
M-30	0.124	0.015
M-30-HD	0.137	0.017
Urban1	0.211	0.029

Table 6
The sample list of detected vehicles and the predicted classes in the Leeds surveillance video, their actual size (Length, Width, Height), and mean absolute percentage error (MAPE) of the estimated 3D bounding box size.

Vehicle name	Predicted Class	Actual size (L, W, H) m	MAPE (%)
Toyota Aygo	Car	$3.4 \times 1.6 \times 1.4$	19.7
Mini Cooper	Car	$3.8 \times 1.7 \times 1.4$	14.6
Vauxhall Corsa	Car	$4.0 \times 1.7 \times 1.4$	12.0
Toyota Yaris	Car	$4.1 \times 1.7 \times 1.6$	14.5
BMW 321I	Car	$4.2 \times 1.3 \times 1.3$	15.2
Ford Focus	Car	$4.4 \times 1.8 \times 1.5$	10.3
Ford Transit	Van	$4.9 \times 1.3 \times 2.0$	18.4
Nissan Navara	Pickup	$5.1 \times 1.8 \times 1.7$	16.1
Ford Ranger	Pickup	$5.3 \times 1.7 \times 1.8$	15.0
Volvo HGV	Articulated	$7.5 \times 2.5 \times 2.5$	10.4
Wright Streetdeck	Bus	$10.5 \times 2.5 \times 4.4$	10.1
Wright HAWK	Bus	$10.6 \times 2.5 \times 3.0$	12.3

- **Parked Vehicles**: a set $\mathcal{P}$ contains all vehicles in $\check{\mathbf{I}}$ domain with less than one-meter distance from the road curb ($l^r_{v_i}$), where their temporal speeds (ϑ_{v_i}) over the past minute have been consistently close to zero (less than 1 mph).
- **Speeding Vehicles**: a set $\mathcal{S}$ consists of vehicles in which their speed (ϑ_{v_i}) is more than the speed limit of the road (e.g. 30 mph for Leeds dataset in England).
- **Hi-Risk Pedestrians**: a set $\mathcal{D}$ consists of pedestrians who are walking at close distances (less than a meter) from the non-Parked vehicles $\mathcal{V}$.

To analyze the traffic condition, we buffer the count of tracked vehicles and pedestrians locations during a period of time (e.g. 6000 frames) as shown by the line graph in Fig. 13(a).

In order to visualize a long-term spatio-temporal statistical analysis of traffic flow and interactions between road users, a heatmap representation is created similar to our previous work in another context for social distancing monitoring (Rezaei & Azarmi, 2020). The heatmap is defined by the matrix $\check{\mathbf{H}}^t \in \mathbb{R}^{\check{w} \times \check{h}}$ in the satellite domain, where t is the frame-ID number. The matrix is initially filled with zero to save the last location of objects using the input image sequences. The heatmap updates in each frame by the function $G_{(\text{object})}(\check{\mathbf{H}})$ using a 3×3 Gaussian matrix centered at the object's location $(\check{x}, \check{y})$ on the $\check{\mathbf{H}}$ matrix. The heatmap intensity values are normalized between 0 and 255 to visualize it as a color-coded heat image. The red spectrum represents the higher values, and the blue spectrum represents the low values.

Figs. 14 and 15 show sample perspective heatmaps of vehicles and pedestrians for GRAM-RTM and UA-DETRAC datasets, respectively.

We analyze the traffic flow in four categories as follows:

4.2.1. Pedestrian and vehicle activities

The heatmap of the detected pedestrians is shown by $\check{\mathbf{H}}_{(p)}$, which updates over time:

$$\check{\mathbf{H}}^t_{(p)} = G_{(p_i)}(\check{\mathbf{H}}^{t-1}_{(p)}) \qquad \forall p_i \in P \tag{16}$$

Fig. 13(b) illustrates the developed heatmap $\check{\mathbf{H}}_{(p)}$ on the satellite image. The lower range values have been removed for better visualization. The figure can provide valuable information about the pedestrians' activity. For instance, although a significant number of pedestrians have crossed the dedicated zebra crossing in this scene, a higher percentage of them have crossed the road in another region (marked by a red rectangle) where there is no zebra crossing. Also, there are a few pedestrians who have crossed the street directly in front of the bus station.

Similarly, the heatmap for detected vehicles is defined as follows:

$$\check{\mathbf{H}}^t_{(v)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(v)}) \qquad \forall v_i \in V \ , \ v_i \notin \mathcal{P} \tag{17}$$

where $\check{\mathbf{H}}_{(v)}$ stores the location of moving vehicles only (not stationary or parked vehicles), as per Fig. 13(c). This heatmap represents that more vehicles are traversing on the left lane of the road compared to the opposite direction, on the right lane.

The heatmap images can be also mapped to the perspective space by: $\hat{\mathbf{H}} = \Lambda(\check{\mathbf{H}}, \mathbf{G}^{-1})$. Figs. 13(d) and 13(e) are corresponding maps of Figs. 13(b) and 13(c), from the perspective view.

4.2.2. Speeding zones

We also investigated the speed violation heatmap $\check{\mathbf{H}}_{(\vartheta)}$ and the areas in which vehicles violated the speed limit of the road:

$$\check{\mathbf{H}}^t_{(\vartheta)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(\vartheta)}) \qquad \forall v_i \in \mathcal{S} \tag{18}$$

Figs. 16(a) and 16(b), illustrate an instance of a speeding heatmap calculated over the 10,000 frames. As can be seen and expected, the speeding violation significantly decreases near the pedestrian crossing zone. As a very useful application of the developed model, similar investigations can be conducted in various parts of city and urban areas, in order to identify less known or hidden hazardous zones where the vehicles may breach the traffic rules.

The graph shown in Fig. 16(c), represents the average speed of all vehicles in the scene during the selected period of the monitoring. In each frame, the average speed is calculated by:

$$\bar{\vartheta} = \frac{\sum \vartheta_{v_i}}{n_v} \qquad \forall v_i \in V, v_i \notin \mathcal{P} \tag{19}$$

where n_v is the number of vehicles that are not in the 'Parked' status.

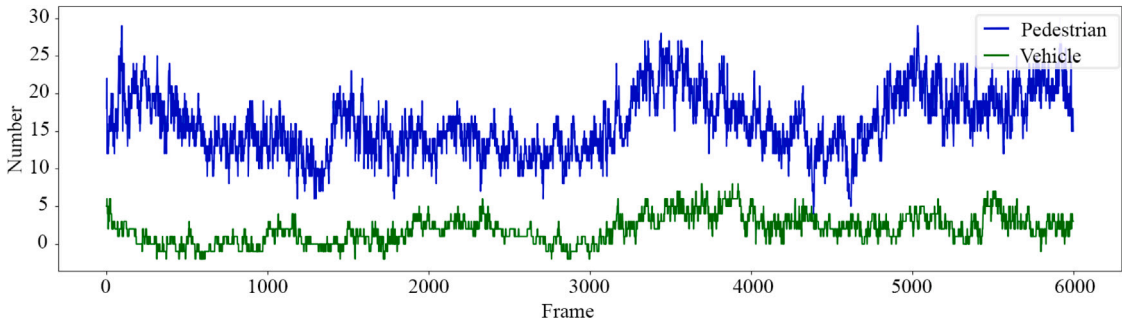
To the best of our knowledge no research has been conducted to monitor the live heatmap of the vehicle and pedestrian interactions using a single camera, applicable to all camera setups thanks to the proposed auto-calibration technique.

4.2.3. Traffic congestion

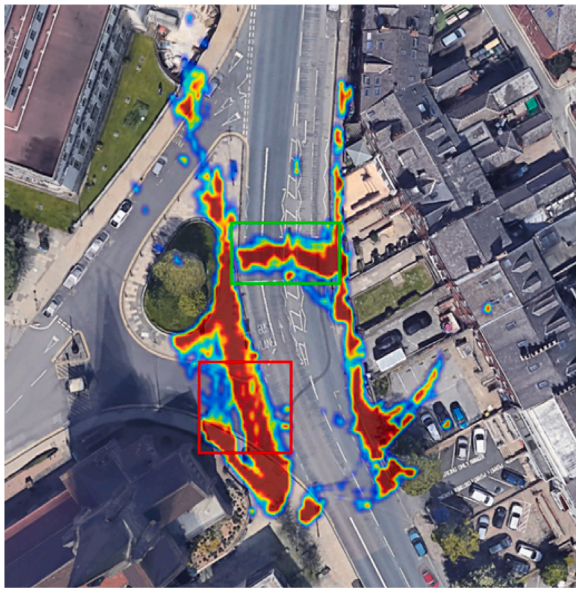
Similarly, in order to identify the congested and crowded spots in the scene, we can monitor the vehicles e.g. with less than 2 m distances to each other with an average speed of e.g. lower than 5 mph. The shorter vehicles' proximity over a longer period of time, the larger values will be stored in the congestion buffer; consequently, a hotter heatmap will be generated. Defining optimum values of distance and speed threshold requires intensive analytical and statistical data collection and assessments based on the road type (e.g. highway or a city road) which is out of the scope of this research. However, as a general-purpose solution and similar to the previous heatmaps, we defined the congestion heatmap $\check{\mathbf{H}}_{(C)}$ as follows:

$$\check{\mathbf{H}}^t_{(C)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(C)}) \qquad \forall v_i \in \mathcal{A} \tag{20}$$

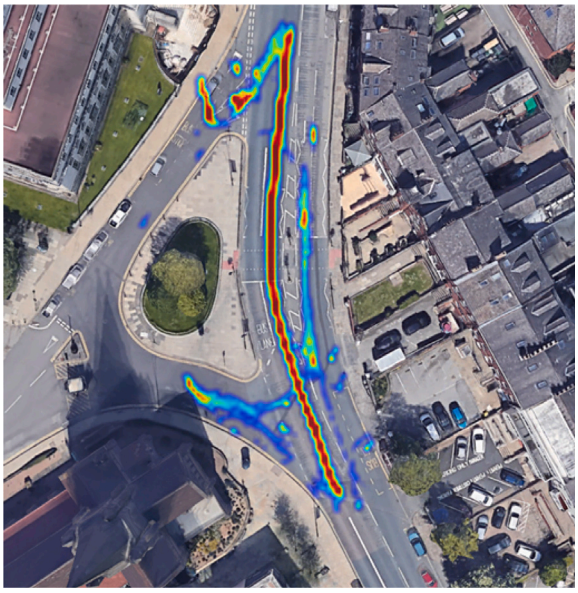
where $\mathcal{A}$ is an ID set of vehicles that are in congested areas. As we can see in Fig. 17, there are two regions of congestion, one before the



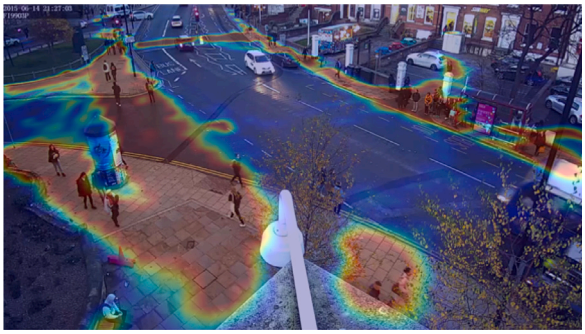
(a) Vehicle and pedestrian counts over 6000 video frames



(b) BEV Pedestrian movements heatmap



(c) BEV vehicle movements heatmap



(d) Pedestrians' perspective view heatmap, same as (b)

(e) Vehicles' perspective view heatmap, same as (c)

Fig. 13. Spatio-temporal long-term analysis of vehicles and pedestrians' activity in Leeds video dataset.

(a) Vehicle movements heatmap for Urban1 video

(b) Pedestrian movements heatmap for Urban1 video

Fig. 14. The analysis results for GRAM Road-Traffic Monitoring video footage.

(a) Vehicle commuting heatmap at footage MVI 40902 (b) Pedestrian commuting heatmap at footage MVI 40902

(c) Vehicle commuting heatmap at footage MVI 40743 (d) Pedestrian commuting heatmap at footage MVI 40743

(e) Vehicle commuting heatmap at footage MVI 40775 (f) Pedestrian commuting heatmap at footage MVI 40775

Fig. 15. The analysis results for UA-DETRAC (Wen et al., 2020) datasets video footage. The Left and Right columns are the heatmap representation of vehicles' and pedestrians' movements, respectively.

pedestrian crossing which is probably due to the red traffic light which stops the vehicles, and also a second congestion spot at the T-junction (top left side of the scene), where the vehicles stop and line up before joining the main road.

4.2.4. Traffic anomaly assessment

Fig. 18 shows the risky pedestrian behavior's heatmap by monitoring the pedestrians who are not maintaining a minimum safety distance of $2m$ to the passing vehicles. The heatmap of the high-risk pedestrians can be updated according to the following equation:

$$\check{\mathbf{H}}^{t}_{(\mathcal{W})} = G_{(p_i)}(\check{\mathbf{H}}^{t-1}_{(\mathcal{W})}) \qquad \forall p_i \in \mathcal{D} \tag{21}$$

The hot area in front of the bus station is more likely caused by the buses which stop just beside the bus station. The heatmap also shows another very unsafe and risky spot in the same scene where some of the pedestrians have crossed through the middle of a complex 3-way intersection. This may have been caused by careless pedestrians who try to reach the Bus Stop or leave the Bus Stop via a high-risk shortcut.

All experiments in section experimental were conducted on a PC workstation with an Intel © Core™ Core i7 12900K processor and an Nvidia RTX 5000 GPU with CUDA version 11. All services were performed based on a unified software using parallel processing for simultaneous utilization of all processor's cores to enhance the execution performance. Similarly, all image-processing-related calculations were performed on GPU tensor units to increase speed and efficiency. The running time of the whole service is 0.05 ms, except for the speed of the object detector which can slightly vary depending on the lighting and complexity of the environment.

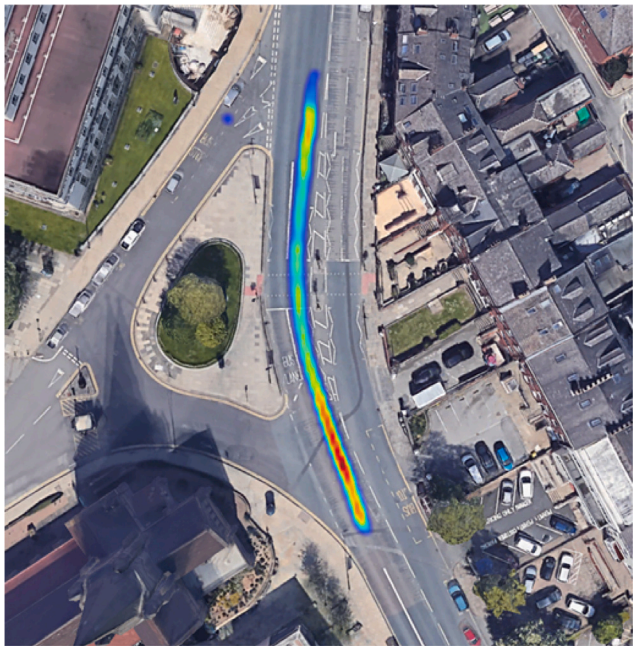
5. Conclusion

The article introduced 3D-Net, an efficient 3D object recognition, and traffic modeling system, to localize and categorize various types of road users. We also developed MCOT (A multi-class object tracker) for the accurate localization and prediction of the future position of road users (vehicles and pedestrians) with significant long-term occlusions and noise, as one of the challenges in the studied application.

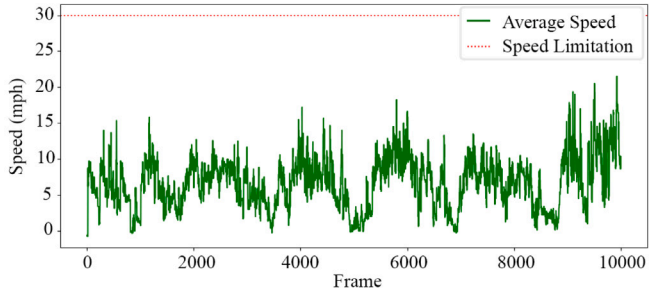
A novel auto-camera calibration technique (called SG-IPM), applicable to the majority of surveillance cameras in urban areas was also introduced to estimate real-world positions and distances of road users using three inputs of satellite images, ground images, and GPS location of the camera.

We demonstrated the combined capability of the proposed detection, tracking, and auto-calibration to accurately approximate the camera parameters, eliminate the perspective effect, and ultimately for traffic analysis. We showed tracking and camera calibration (in particular), play a critical role in traffic monitoring. Any small positioning shift due to miscalibration, or weak tracking provides false information about traffic status in a particular lane or part of the road scene. Thanks to the accurate calibration and tracking algorithm, we

(a) Speed violation heatmap - Perspective view



(b) Speed violation heatmap - Bird's eye view



(c) Average speed of moving vehicles in the scene

Fig. 16. Automated speed monitoring and heatmap analysis based on 10,000 video frames from Leeds dataset.

are able to collect comprehensive information about traffic volume and the vehicles traveling in each road lane (as evident in Figs. 13–15).

Using real-time positioning and spatio-temporal tracking information, the heading angle of objects was also calculated. The ABF method helped to remove the sudden angle variation noise, due to occlusion, sensor limitation, or detection imperfection.

Fig. 17. Heatmap representation of congested areas based on 10,000 live video frames from Leeds dataset.

Fig. 18. Heatmap representation of areas in which vehicles and pedestrians were too close to each other.
Source: 10,000 live video frames from Leeds dataset.

The above three models enabled us to conduct a real-world experiment in the context of Intelligent Transportation Systems, by performing 3D bounding box estimation and traffic heatmap modeling and analysis. This helps the researchers, and stakeholders to better understand and analyze road status, drivers tailgating, congestion, high-risk areas, and pedestrian–vehicle interactions in order to ensure safer and more resilient transportation and to plan for future mobility.

Experimental results on the MIO-TCD dataset and a real-world roadside camera proved the proposed approach dominates 19 state-of-the-art research works in ten categories of vehicles and pedestrian detection. Accurate tracking, auto-calibration, and automated congestion detection with a high level of accuracy (up to 84.6%) and stability over various lighting conditions were other contributions of this research.

CRediT authorship contribution statement

Mahdi Rezaei: Conceptualization, Methodology, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Mohsen Azarmi:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Farzam Mohammad Pour Mir:** Conceptualization, Methodology, Software, Writing – original draft, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and data that support the findings of this study are openly available in Code Ocean capsule and data repository at https://codeocean.com/capsule/7713588/tree.

Acknowledgments

The authors would like to thank all partners within the Hi-Drive project for their cooperation and valuable contribution. This research has received funding from the European Union's Horizon 2020 research and innovation programme, under grant Agreement No 101006664. The article reflects only the author's view and neither the European Commission nor CINEA is responsible for any use that may be made of the information this document contains.

Appendix A. Camera calibration and inverse perspective mapping

Knowing the camera's intrinsic and extrinsic parameters, the actual position of the 3D objects from a 2D perspective image can be estimated using Inverse Perspective Mapping (IPM) as follows:

$$[x \quad y \quad 1]^T = \mathbf{K}[\mathbf{R}|\mathbf{T}][X_w \quad Y_w \quad Z_w \quad 1]^T \tag{22}$$

where x and y are the pixel coordinates of the image, X_w, Y_w and Z_w are coordinates of points in real world. $\mathbf{K}$ is the camera intrinsic matrix:

$$\mathbf{K} = \begin{bmatrix} f * k_x & s & c_x & 0 \\ 0 & f * k_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{23}$$

where f is the focal length of the camera, k_x and k_y are the calibration coefficient values in horizontal and vertical pixel axis, s is the shear coefficient and (c_x, c_y) are the principal points shifting the optical axis of the image plane. $\mathbf{R}$ is the rotation matrix:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_c & -\sin\theta_c & 0 \\ 0 & \sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

where θ_c is the camera angle.

$\mathbf{T}$ is the translation matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{h_c}{\sin\theta_c} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

where h_c is the height of the camera.

These three matrices together makes the projection matrix $\mathbf{G} \in \mathbb{R}^{3\times4}$ as follows:

$$\mathbf{G} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{26}$$

so, the transformation equation can be summarized as $[x \quad y \quad 1]^T = \mathbf{G}\,[X_w \quad Y_w \quad Z_w \quad 1]^T$.

Assuming the camera is looking perpendicular to the ground plane of the scene, the Z_w parameter is removed. A reduction in the $\mathbf{G}$ matrix size, turns it into a planar transformation matrix $\mathbf{G} \in \mathbb{R}^{3\times3}$ with g_{ij} elements as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix} \tag{27}$$

Therefore, for every pixel point (x, y), the planar transformation function can be represented as follow:

$$\Lambda((x,y),\mathbf{G}) = \left(\frac{g_{11}\times x + g_{12}\times y + g_{13}}{g_{31}\times x + g_{32}\times y + g_{33}}, \frac{g_{21}\times x + g_{22}\times y + g_{23}}{g_{31}\times x + g_{32}\times y + g_{33}}\right) \tag{28}$$

Appendix B. Bird's eye view tracker

We model the vehicles' maneuver and movement tracking using a constant velocity Kalman filter (Bewley et al., 2016). The system state x is a 1×6 vector, which contains locations of objects $(\check{x}, \check{y})$ in bird's eye view coordinates. It is defined as:

$$x = [\check{x}, 0, 0, \check{y}, 0, 0]^T \tag{29}$$

The state update equation is given by:

$$x_{n,n} = x_{n,n-1} + K_n(z_n - H x_{n,n-1}) \tag{30}$$

where $x_{n,n}$ is the estimated system state vector at time step n and $x_{n,n-1}$ is the predicted system state vector at time step $n-1$, H is a 2×6 observation matrix in which all elements are zero, except $h_{1,0}$ and $h_{2,4}$ which are equal to 1, to provide a measurement on $\check{x}, \check{y}$ coordinates of the objects, $z_n = H x_n$ is the measurements, where x_n is the true system state (hidden state); and K is the Kalman Gain matrix as follows:

$$K_n = P_{n,n-1} H^T (H P_{n,n-1} H^T + R_n)^{-1} \tag{31}$$

where n represents the time step, $P_{n,n-1}$ is a prior estimate uncertainty (covariance) matrix of the current state which is predicted at the previous state; R_n is the covariance of the observation noise which is defined as follows:

$$R_n = \begin{bmatrix} \sigma_{\check{x}}^2 & 0 \\ 0 & \sigma_{\check{y}}^2 \end{bmatrix} \tag{32}$$

where $\sigma_{\check{x}}, \sigma_{\check{y}}$ are the measurement error standard deviation for the projected on BEV coordinates, and they are equal to $\sqrt{\sigma_d^2 + \sigma_s^2}$ (based on the given values in Section 4).

The estimated uncertainty matrix at time step n is given by the covariance exploration equation:

$$P_{n+1,n} = F P_{n,n} F^T + Q \tag{33}$$

where $P_{n,n}$ is the uncertainty (covariance) matrix of the current state estimation, which is a 6×6 diagonal matrix with an initial value of 100, to consider high uncertainty for the initial state of the system, and $P_{n+1,n}$ is the uncertainty (covariance) matrix of the next state estimation (prediction), F is the state transition model, and Q is process noise.

We define the matrix F for the constant velocity tracking as follows:

$$F = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

where Δt is the measurement period and it is equal to 1 s. We also define process noise matrix Q as:

$$Q = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{35}$$

where σ_a is the random acceleration standard deviation and it is equal to 0.04 in our dataset.

References

Abdollahi, M., Khaleghi, T., & Yang, K. (2020). An integrated feature learning approach using deep learning for travel time prediction. *Expert Systems with Applications, 139*, Article 112864.

Ahmad, A. (2021). *Assessment of automated road features xtraction algorithm from UAV images* (Ph.D. thesis), Universiti Teknologi Mara Perlis.

Alldieck, T., Bahnsen, C. H., & Moeslund, T. B. (2016). Context-aware fusion of RGB and thermal imagery for traffic monitoring. *Sensors*, *16*(11).

Arinaldi, A., Pradana, J. A., & Gurusinga, A. A. (2018). Detection and classification of vehicles for traffic video analytics. *Procedia Computer Science*, *144*, 259–268, INNS Conference on Big Data and Deep Learning.

Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3782–3795.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing*. IEEE.

Bhoi, A. (2019). Monocular depth estimation: A survey. arXiv:1901.09402.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, *300*, 17–33.

Bui, K.-H. N., Yi, H., & Cho, J. (2020). A multi-class multi-movement vehicle counting framework for traffic analysis in complex areas using CCTV systems. *Energies*, *13*(8).

Chen, K., Chen, F., Lai, B., Jin, Z., Liu, Y., Li, K., et al. (2020). Dynamic spatio-temporal graph-based CNNs for traffic flow prediction. *IEEE Access*, *8*, 185136–185145.

Chen, C., Li, K., Teo, S. G., Chen, G., Zou, X., Yang, X., et al. (2018). Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In *2018 IEEE international conference on data mining* (pp. 893–898). IEEE.

Cheon, M., Lee, W., Yoon, C., & Park, M. (2012). Vision-based vehicle detection system with consideration of the detecting location. *IEEE Transactions on Intelligent Transportation Systems*, *13*(3), 1243–1252.

Cheung, S.-c. S., & Kamath, C. (2004). *vol. 5308*, *Robust techniques for background subtraction in urban traffic video, vol. 5308* (pp. 881–892). SPIE,

Chintalacheruvu, N., Muthukumar, V., et al. (2012). Video based vehicle detection and its application in intelligent transportation systems. *Journal of Transportation Technologies*, *2*(04), 305.

Cui, Z., Henrickson, K., Ke, R., & Wang, Y. (2019). Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *21*(11), 4883–4894.

Dubská, M., Herout, A., Juránek, R., & Sochor, J. (2015). Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, *16*(3), 1162–1171.

Dutta, A., Mondal, A., Dey, N., Sen, S., Moraru, L., & Hassanien, A. E. (2020). Vision tracking: A survey of the state-of-the-art. *SN Computer Science*, *1*(1), 1–19.

Fernandes, D., Silva, A., Névoa, R., Simões, C., Gonzalez, D., Guevara, M., et al. (2021). Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy. *Information Fusion*, *68*, 161–191.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Gawande, U., Hajari, K., & Golhar, Y. (2020). Pedestrian detection and tracking in video surveillance system: Issues, comprehensive review, and challenges. *Recent Trends in Computational Intelligence*.

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOx: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430.

Guerrero-Gomez-Olmedo, R., Lopez-Sastre, R. J., Maldonado-Bascon, S., & Fernandez-Caballero, A. (2013). Vehicle tracking by simultaneous detection and viewpoint estimation. In *IWINAC 2013, Part II, LNCS 7931* (pp. 306–316).

Hedeya, M. A., Eid, A. H., & Abdel-Kader, R. F. (2020). A super-learner ensemble of deep networks for vehicle-type classification. *IEEE Access*, *8*, 98266–98280.

Hu, W., Tan, T., Wang, L., & Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *34*(3), 334–352.

Huang, X., Wang, X., Lv, W., Bai, X., Long, X., Deng, K., et al. (2021). PP-YOLOv2: A practical object detector. arXiv preprint arXiv:2104.10419.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., et al. (2019). A survey of deep learning-based object detection. *IEEE Access*, *7*, 128837–128868.

Jung, H., Choi, M.-K., Jung, J., Lee, J.-H., Kwon, S., & Young Jung, W. (2017). ResNet-based vehicle classification and localization in traffic surveillance systems. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) workshops*.

Kashyap, A. A., Raviraj, S., Devarakonda, A., Nayak K, S. R., KV, S., & Bhat, S. J. (2022). Traffic flow prediction models–A review of deep learning techniques. *Cogent Engineering*, *9*(1), Article 2010510.

Kim, Z. (2009). Camera calibration from orthogonally projected coordinates with noisy-RANSAC. In *2009 workshop on applications of computer vision* (pp. 1–7).

Laga, H. (2019). A survey on deep learning architectures for image-based depth reconstruction. arXiv:1906.06113.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *2017 IEEE international conference on computer vision* (pp. 2999–3007).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755). Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37). Springer.

Luo, Z., Branchaud-Charron, F., Lemaire, C., Konrad, J., Li, S., Mishra, A., et al. (2018). MIO-TCD: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, *27*(10), 5129–5141.

Mallikarjuna, C., & Rao, K. R. (2011). Heterogeneous traffic flow modelling: a complete methodology. *Transportmetrica*, *7*(5), 321–345.

Mandal, V., Mussah, A. R., Jin, P., & Adu-Gyamfi, Y. (2020). Artificial intelligence-enabled traffic monitoring system. *Sustainability*, *12*(21), 9177.

Min Gan, H., Fernando, S., & Molina-Solana, M. (2021). Scalable object detection pipeline for traffic cameras: Application to Tfl JamCams. *Expert Systems with Applications*, *182*, 115–154.

Nambiar, R., Shroff, R., & Handy, S. (2018). Smart cities: Challenges and opportunities. In *2018 10th international conference on communication systems networks* (pp. 243–250).

Niu, H., Lu, Q., & Wang, C. (2018). Color correction based on histogram matching and polynomial regression for image stitching. In *2018 IEEE 3rd international conference on image, vision and computing* (pp. 257–261).

Olatunji, I. E., & Cheng, C.-H. (2019). Video analytics for visual surveillance and applications: An overview and survey. In *Machine learning paradigms: Applications of learning and analytics in intelligent systems Machine Learning Paradigms*, 475–515.

Oliveira, M., Santos, V., & Sappa, A. D. (2015). Multimodal inverse perspective mapping. *Information Fusion*, *24*, 108–121.

Peppa, M. V., Komar, T., Xiao, W., James, P., Robson, C., Xing, J., et al. (2021). Towards an end-to-end framework of CCTV-based urban traffic volume detection and prediction. *Sensors*, *21*(2).

Poddar, M., Giridhar, M., Prabhu, A. S., Umadevi, V., et al. (2016). Automated traffic monitoring system using computer vision. In *2016 international conference on ICT in business industry & government* (pp. 1–5). IEEE.

Rezaei, M., & Azarmi, M. (2020). Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic. *Applied Sciences*, *10*(21), 7514.

Rezaei, M., & Klette, R. (2017). *Computer vision for driver assistance, vol. 45*. Cham: Springer International Publishing.

Rezaei, M., Terauchi, M., & Klette, R. (2015). Robust vehicle detection and distance estimation under challenging lighting conditions. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2723–2743.

Sheng, H., Yao, K., & Goel, S. (2021). Surveilling surveillance: Estimating the prevalence of surveillance cameras with street view data. http://dx.doi.org/10.1007/978-3-642-38622-0_32, arXiv preprint arXiv:2105.01764.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820.

Sochor, J., Juránek, R., & Herout, A. (2017). Traffic surveillance camera calibration by 3D model bounding box alignment for accurate vehicle speed measurement. *Computer Vision and Image Understanding*, *161*, 87–98.

Song, H., Li, C., Wu, F., Dai, Z., Wang, W., Zhang, Z., et al. (2019). 3D vehicle model-based PTZ camera auto-calibration for smart global village. *Sustainable Cities and Society*, *46*, Article 101401.

Song, Y., Yao, J., Ju, Y., Jiang, Y., & Du, K. (2020). Automatic detection and classification of road, car, and pedestrian using binocular cameras in traffic scenes with a common framework. *Complexity*, *2020*.

Tan, M., Pang, R., & Le, Q. V. (2019). EfficientDet: Scalable and efficient object detection. In *2020 IEEE/CVF conference on computer vision and pattern recognition* (pp. 10778–10787).

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. http://dx.doi.org/10.48550/arXiv.2207.02696, arXiv abs/2207.02696.

Wang, T., He, X., Su, S., & Guan, Y. (2017). Efficient scene layout aware object detection for traffic surveillance. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) workshops*.

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 390–391).

Wang, S.-H., Tu, C.-H., & Juang, J.-C. (2021). Automatic traffic modelling for creating digital twins to facilitate autonomous vehicle development. *Connection Science*, 1–20.

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2021). You only learn one representation: Unified network for multiple tasks. arXiv preprint arXiv:2105.04206.

Wang, Y., Zhao, M., Yu, X., Hu, Y., Zheng, P., Hua, W., et al. (2022). Real-time joint traffic state and model parameter estimation on freeways with fixed sensors and connected vehicles: State-of-the-art overview, methods, and case studies. *Transportation Research Part C (Emerging Technologies)*, *134*, Article 103444.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M., Qi, H., et al. (2020). UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2. https://github.com/facebookresearch/detectron2.

Xie, J., Girshick, R., & Farhadi, A. (2016). Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European conference on computer vision* (pp. 842–857). Springer.

Xu, S., Wang, X., Lv, W., Chang, Q., Cui, C., Deng, K., et al. (2022). PP-YOLOE: An evolved version of YOLO. arXiv preprint arXiv:2203.16250.

Yang, W., Fang, B., & Tang, Y. Y. (2018). Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(5), 755–766.

Yang, J., Liu, S., Li, Z., Li, X., & Sun, J. (2022). Real-time object detection for streaming perception. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5385–5395).

Yu, G., & Morel, J.-M. (2011). ASIFT: An algorithm for fully affine invariant comparison. *Image Processing on Line*, *1*, 11–38.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334.

Zhang, J., Xiao, W., Coifman, B., & Mills, J. P. (2020). Vehicle tracking and speed estimation from roadside lidar. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 5597–5608.

Zhang, Z., Zheng, J., Xu, H., & Wang, X. (2019). Vehicle detection and tracking in complex traffic circumstances with roadside LiDAR. *Transportation Research Record*, *2673*(9), 62–71.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34* (pp. 12993–13000).

Zhou, T., Fan, D.-P., Cheng, M.-M., Shen, J., & Shao, L. (2021). RGB-D salient object detection: A survey. *Computational Visual Media*, *7*, 1–33.

Zhou, J., Gao, D., & Zhang, D. (2007). Moving vehicle detection for automatic traffic monitoring. *IEEE Transactions on Vehicular Technology*, *56*(1), 51–59.