Traffic-Net: 3D Traffic Monitoring Using a Single Camera

Mahdi Rezaei [1,⋆,✉], **Mohsen Azarmi** [2,⋆], **Farzam Mohammad Pour Mir** [3,⋆]

[1] Institute for Transport Studies, The University of Leeds, Leeds, LS2 9JT, UK
[2] Department of Computer Engineering, Qazvin University, Qazvin, IR
[3] Faculty of Computer Engineering, Tehran Azad University, Science & Research Branch, IR
[1] m.rezaei@leeds.ac.uk [2] m.azarmi@qiau.ac.ir [3] f.mohammadpour@srbiau.ac.ir

ABSTRACT

Computer Vision has played a major role in Intelligent Transportation Systems (ITS) and traffic surveillance. Along with the rapidly growing automated vehicles and crowded cities, the automated and advanced traffic management systems (ATMS) using video surveillance infrastructures have been evolved by the implementation of Deep Neural Networks. In this research, we provide a practical platform for real-time traffic monitoring, including 3D vehicle/pedestrian detection, speed detection, trajectory estimation, congestion detection, as well as monitoring the interaction of vehicles and pedestrians, all using a single CCTV traffic camera. We adapt a custom YOLOv5 deep neural network model for vehicle/pedestrian detection and an enhanced SORT tracking algorithm. For the first time, a hybrid satellite-ground based inverse perspective mapping (SG-IPM) method for camera auto-calibration is also developed which leads to an accurate 3D object detection and visualisation. We also develop a hierarchical traffic modelling solution based on short- and long-term temporal video data streams to understand the traffic flow, bottlenecks, and risky spots for vulnerable road users. Several experiments on real-world scenarios and comparisons with state-of-the-art are conducted using various traffic monitoring datasets, including MIO-TCD, UA-DETRAC and GRAM-RTM collected from highways, intersections, and urban areas under different lighting and weather conditions.

Keywords – 3D Object Detection; Traffic Flow Monitoring; Intelligent Transportation Systems; Deep Neural Networks; Vehicle Detection; Pedestrian Detection; Inverse Perspective Mapping Calibration; Digital Twins; Video Surveillance.

1 Introduction

Smart video surveillance systems are becoming a common technology for traffic monitoring and congestion management. Parallel to the technology 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 pedestrians and road users [1]. The rapidly growing number of surveillance cameras (over 20 million CCD cameras only in USA and UK) in the arteries of cities, crowded places, roads, intersections, and highways, demonstrator the importance of video surveillance for city councils, authorities and governments [2]. A large network of interconnected surveillance cameras can provide a special platform for further studies on traffic management and urban planning [3]. However, in such a dense and complex road environments, the conventional monitoring of road condition is a very tedious, time-consuming, yet less accurate approach than automated computer vision and AI-based solutions. Hence, automated video surveillance has been researched for many years to gradually replace the humans with computers that can analyse the live traffic and provide effective solutions to maintain transportation safety and sustainability [4].

Computer Vision is one of the most investigated technologies for automated video surveillance inspired by the human

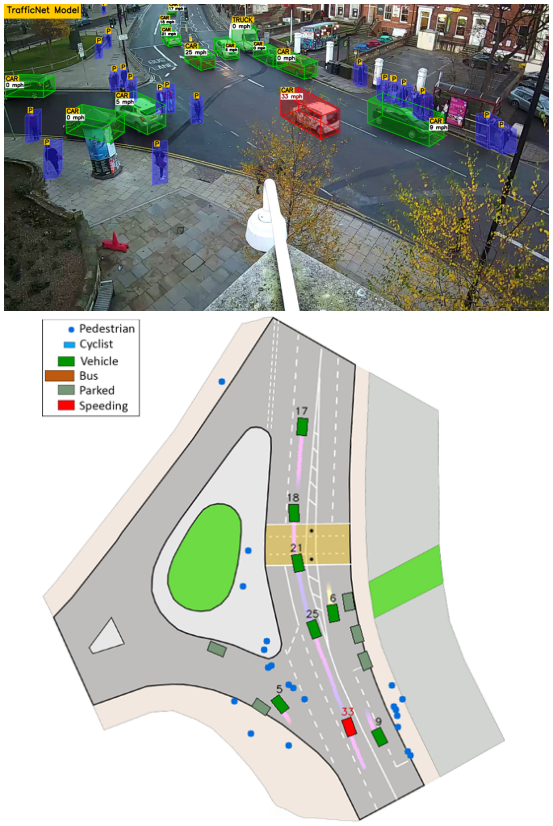


Figure 1. Top: 3D object detection and speed estimation. Bottom: Digital twin and modelling of the same scene.

⋆ The authors contributed equally to this study.
✉ Corresponding Author: m.rezaei@leeds.ac.uk (M. Rezaei)

visual mechanism. The technology aims to enable computers to analyse and interpret the content of digital videos. In automated traffic monitoring systems (ATMS) and Intelligent Transportation Systems (ITS), computer vision can extract a wide range of information from the traffic scenes [5].

Vehicle type recognition, vehicle counting, speed estimation, tracking, and trajectory estimation are examples of automated traffic scene analysis. Figure 1 represents a sample scenario of road complexities including interactions between road users (pedestrians, vehicles, cyclists), moving trajectories, speed detection, and the density of the road users in various points of the road scene. Figure 1, top row shows a 3D road-user detection and localisation, and the bottom row shows the bird's eye view mapping and digital twin modelling of the same scene, after camera calibration and inverse perspective mapping (IPM).

In such highly dynamic environments, the ability of real-time processing and accurate detection of multiple events is crucial [6]. Furthermore, an efficient traffic monitoring system should be capable of working with a grid of various interconnected cameras on different urban locations, where each camera may have a different resolution, viewing angles, height, or focal length. This requires calibration of each and every single camera based on the intrinsic camera parameters and the mounting spec of each camera.

Although various methods of camera calibration such as vanishing-based techniques [7] and multi-point calibrations techniques [8] have been introduced for bird's eye view mapping, fewer investigations have been conducted in the community to introduce automated calibration methods.

The heart of an ATMS is the vehicle and pedestrian identification, and in the field of computer vision, this task is handled by object detection algorithms and tracking techniques [9].

In the past decade, deployment of Deep Neural Networks (DNN) has led to significant advances in indoor object detection. The effectiveness and the accuracy of these contemporary improvements should be investigated for the particular application of traffic monitoring in a complex, dynamic, noisy and crowded environment.

Further challenges such as bad weather conditions, challenging lighting conditions [10] during day and night, as well as occlusion may also affect the performance of the object detection in traffic monitoring systems [11].

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

- Adapting a custom Deep Neural Network (DNN) for vehicle/pedestrian detection.
- Developing an enhanced multi-object and multi-class tracking and trajectory estimation.
- Developing a hybrid satellite/ground-based inverse perspective mapping (SG-IPM) and calibration method for accurate localisation and distance estimation.
- 3D object bounding box estimation of the road users using a single-view camera.
- Automated short- and long-term surveillance solutions to understand traffic bottlenecks, risks, and hazards for road users.

Figure 1 represents a sample output of our contributions including 3D detection, tracking, and environment modelling. Comprehensive details and discussions will be provided in the next sections as follows:

In Section 2 a literature review on both conventional and modern related works is conducted. Section 3 introduces our methodology as an enhanced object detection and tracking algorithm followed by presenting a novel satellite/ground-based auto-calibration technique. In this section, we provide an environment modelling technique as well as the 3D representation of detected objects. Experimental results, evaluations, and comparisons with state-of-the-art will be discussed in Section 4, and finally, Section 5 concludes the article by discussing the challenges and potentials for future works.

2 Related Work

In this section, we review three types of related works to automated traffic surveillance systems (ATMS) including classic object detection methods, modern object detection research directions, and also the CCTV camera calibration solutions, as the prerequisite of any object detection methodology in the context of traffic surveillance. Both classical and machine learning-based methods for automated video surveillance (AVS), automated traffic surveillance systems (ATMS), as well as the camera calibration techniques will be reviewed.

Among classical methods, a series of studies have focused on background subtraction (BGS) techniques for detecting moving objects. Cheung et al. [12] have compared the performance of different BGS methods such as the Mixture of Gaussian (MOG), Median filter (MF), Kalman filter (KF) and frame differentiation (FD) in various weather conditions on a road-side surveillance camera. They reported a higher precision rate using the MOG method. This method estimates various Gaussian distributions that match with the intensity distribution of the image background's content.

Zhao et al. [13] have introduced an adaptive background estimation technique. They divide the image into small none-overlapped blocks followed by the principal component analysis (PCA) on each block's feature. Then they utilise the support vector machine (SVM) to classify the vehicles. The method seems to be robust in partial occlusion and bad illumination conditions. However, it fails to detect stationary objects. The presented system is only evaluated on ideal highway images and neglects the crowded urban roads.

Chintalacheruvu et al. [14] have introduced a vehicle detection and tracking system based on the Harris-Stephen corner detector algorithm. The method focuses on speed violation detection, congestion detection, vehicle counting, and average speed estimation in regions of interest. However, the presented method requires prior road information such as the number of lanes and road directions.

In another approach, Cheon et al. [15] have presented a vehicle detection system using histogram of oriented gradients (HOG) considering the shadow of the vehicles to localise them. They have also used an auxiliary feature vector to enhance the vehicle classification and to find the areas with a high risk of accidents. However, the method leads to erroneous vehicle localisation during the night or day-time where the vehicles' shadows are too long and not presenting the exact location and size of the vehicle.

Although most of the discussed methods perform well in simple and controlled environments, they fail to propose accurate performance in complex and crowded scenarios. Furthermore, they are unable to perform multi-class classifications and can not distinguish between various categories of moving objects such as pedestrians, cars, buses, trucks, etc.

With the emergence of *deep neural networks (DNNs)*, the machine learning domain received more attention in the object detection domain. In modern object detection algorithms, Convolutional Neural Networks (CNN) learns complex features during the training phase, aiming to elaborate and understand the contents of the image. This normally leads to improvement in detection accuracy compared to classical image processing methods [16]. Such 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 RCNN family, consist of a region proposal stage and a classification stage [17]; while the single-stage object detectors such as Single-Shot Multi-Box Detector (SSD) [18], and You Only Look Once (YOLO) see the detection process as a regression problem, thus provide a single-unit localisation and classification architecture [17].

Arinaldi et al. [19] reached a better vehicle detection performance using Faster-RCNN compared to a combination of MOG and SVM models.

Peppa et al. [20], developed a statistical-based model, a random forest method, and an LTSM to predict the traffic volume for the upcoming 30 minutes, to compensate for lack of accurate information in extreme weather conditions.

Some researchers such as Bui et al. [21], utilised single-stage object detection algorithms including he YOLOv3 model for automated vehicle detection. They designed a multi-class distinguished-region tracking method to overcome the occlusion problem and lighting effects for traffic flow analysis.

In [22], Mandal et al. 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.

On the other hand, the advancement of stereo vision sensors and 3D imaging has led to more accurate solutions for traffic monitoring as well as depth and speed estimation for road users. Consequently, this enables the researchers to distinguish the scene background from the foreground objects, and measure the objects' size, volume, and spatial dimensions [23].

LiDAR sensors and 3D point cloud data offers a new mean for traffic monitoring. In [24], Zhang et al. have presented a real-time vehicle detector and tracking algorithm without bounding box estimation, and by clustering the point cloud space. Moreover, they used the adjacent frame fusion technique to improve the detection of vehicles occluded by other vehicles on the road infrastructures.

Authors in [25], proposed a centroid-based tracking method and a refining module to track vehicles and improve the speed estimations. Song, Yongchao, et al. [26] proposed a framework which uses binocular cameras to detect road, pedestrians and vehicles in traffic scenes.

In another multi-modal research, thermal sensor data is fused with the RGB camera sensor, leading to a noise-resistant technique for traffic monitoring [27].

Although many studies are conducting different sensors to perform 3D object detection such as in [28], [29], the cost of applying such methods in large and crowded cities could be significant. Since there are many surveillance infrastructures already installed in urban areas and there are more data available for this purpose, 2D object detection on images has gained a lot of attention in a more practical way.

Many studies including deep learning-based methods [30], [31], have tried to utilise 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 [32].

Regardless of the object detection methodology, the CCTV *camera calibration* is a key requirement of 2D or 3D traffic condition analysis prior to starting any object detection operation. A camera transforms the 3D world scene into a 2D perspective image based on the camera intrinsic and extrinsic parameters. Knowing these parameters is crucial for an accurate inverse perspective mapping, distance estimation, and vehicle speed estimation [33].

In many cases especially when dealing with a large network of CCTV cameras in urban areas, these parameters can be unknown or different to each other due to different mounting setups and different types of cameras. Individual calibration of all CCTVs in metropolitan cities and urban areas with thousands of cameras is a very cumbersome and costly task. Some of the existing studies have proposed camera calibration techniques in order to estimate these parameters, hence estimating an inverse perspective mapping.

Dubska et al. [34] extract vanishing points that are parallel and orthogonal to the road in a road-side surveillance camera image, using the moving trajectory of the detected cars and Hough line transform algorithm. This can help to automatically calibrate the camera for traffic monitoring purposes, despite low accuracy of the Hough transform algorithm in challenging lighting and noisy conditions.

Authors in [35], proposed a Faster-RCNN model to detect vehicles and consider car edgelets to extract perpendicular van-

ishing points to the road to improve the automatic calibration of the camera.

Song et al. [36], have utilised 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. Then, they consider a fixed average length, width and height of cars to draw 3D bounding boxes.

However, all of the aforementioned calibration methods assume 1) The road has zero curvature which is not the case in real-world scenarios and 2) Vanishing points are based on straight-line roads. i.e. the model does not work on intersections.

In a different approach, Kim et al. [37], consider 6 and 7 corresponding coordinates in a road-side camera image and an image with a perpendicular view of the same scene (such as near-vertical satellite image) to automatically calibrate the camera. They introduced a revised version of RANSAC model, called Noisy-RANSAC to efficiently work with at least 6 or 7 corresponding points produced by feature matching methods. However, the method is not evaluated on real-world and complex scenarios in which the road is congested and occluded by various types of road users.

Among the reviewed literature most of the studies have not investigated various categories of road users such as pedestrians and different types of vehicles that may exist in the scene. Moreover, there are limited researches addressing full/partial occlusion challenges in the congested and noisy environment of urban areas. It is also notable that the performance of the latest object detection algorithm to date is not evaluated by the traffic monitoring related researches.

Furthermore, very limited research has been conducted on short and long-term spatio-temporal video analysis to automatically understand the interaction of vehicles and pedestrians and their effects on the traffic flow, congestion, hazards, or accidents.

In this article, we will aim at proving an efficient and estate-of-the-art traffic monitoring solution to tackle some of above-mentioned research gaps and weaknesses in congested urban areas.

3 Methodology

We represent our methodology in four hierarchical subsections. In section 3.1 as the first contribution, a customised and highly accurate vehicle and pedestrian detection model will be introduced. In Section 3.2 and as the second contribution we elaborate our multi-object and multi-class tracker (MOMCT). Next, in Section 3.3, a novel auto-calibration technique (named SG-IPM) is developed. Last but not the least, in section 3.4 we develop a hybrid methodology for road and traffic environment modelling which leads to 3D detection and representation of all vehicles and pedestrians in a road scene using a single CCTV camera.

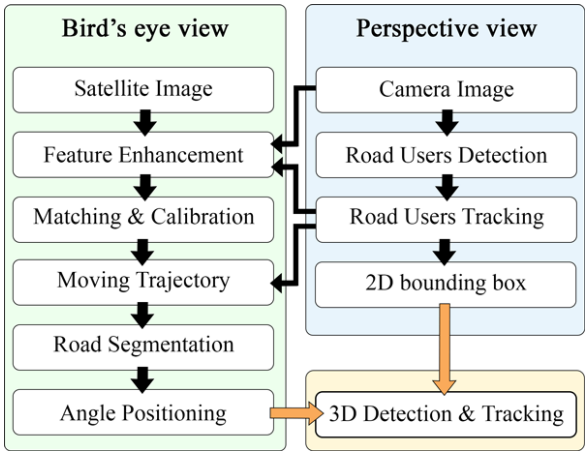


Figure 2. The overall structure of the proposed methodology

Figure 2 summarises the overall flowchart of the methodology, starting with a 2D camera image and satellite image as the main inputs which ultimately lead to a 3D road-users detection and tracking.

3.1 Object Detection and Localisation

According to the reviewed literature, the YOLO family has been proven to be faster and also very accurate compared to most of the state-of-the-art object detectors [38].

We hypothesis that recent versions of the YOLO family can provide a balanced trade-off between the speed and accuracy of our traffic surveillance application. In this section we conduct a domain adaptation and transfer learning of YOLOv5. The Microsoft COCO dataset [39] consists of 80 annotated categories of the most common indoor and outdoor objects in daily life. We use pre-trained feature extraction matrices of YOLOv5 model on COCO dataset as the initial weights to train our customised model.

Our adapted model is designed to detect 11 categories of traffic-related objects which also match the MIO-TCD traffic monitoring dataset [40]. These categories consist of a pedestrian class and 10 types of vehicles, including articulated truck, bicycle, bus, car, motorcycle, motorised vehicles, non-motorised vehicles, pickup truck, single-unit truck and work van. Because of the different number of classes in two datasets, the last layers of the model (the output layers) do not have the same shape to copy. Therefore, these layers will be initialised with random weights (using 100100 seeds). After the initialisation process, the entire model will be trained on the MIO-TCD dataset. We expect this would ultimately lead to a more accurate and customised model for our application.

As shown in Figure 3 the architecture of modern YOLO frameworks consists of a *backbone*, the *neck*, and the *head*.

The backbone includes stacked convolutional layers turning the input into the feature space. In the backbone, the Cross-Stage Partial network (CSP) [41] conducts shortcut connections between layers to improve the speed and accuracy of feature extractors.

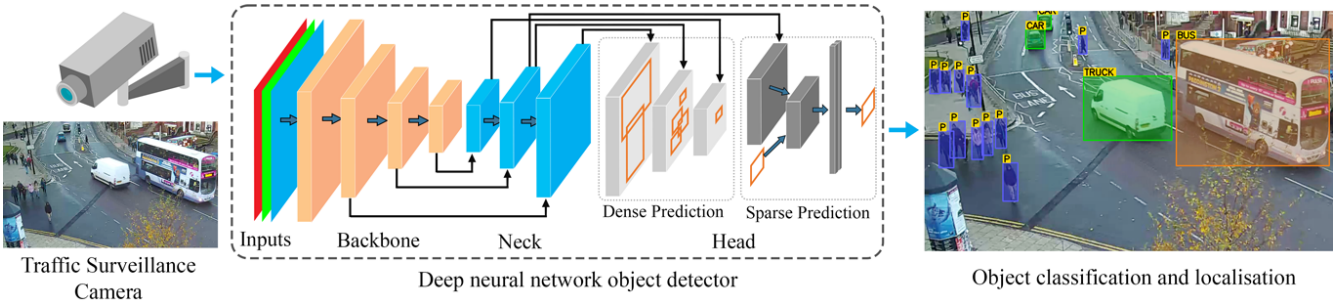


Figure 3. Summarised structure of the dense (single-stage) and sparse (two-stage) object detection architecture, applied to a road-side surveillance video.

The neck consists of feature enhancers such as Spacial Pyramid Pulling (SPP) [42], and Path Aggregation Network (PAN) [43], concatenating the features extracted from initial layers (closer to the input of the model) with the end layers (closer to the head of the model) to enhance the semantic and spacial information and to improve the detection accuracy for small objects.

The head part of YOLOv5 performs convolutional operations on the enhanced features to generate outputs (predictions) with different scales. The count and size of the outputs vary based on the number of pre-defined multi-scale grid cells, anchor boxes, and also the number of ground truth classes. The anchor boxes are determined based on scales of the existing bounding boxes in the ground truth using the k-means clustering algorithm. The model optimises Focal-Loss [44] and Distance-IoU loss [45] functions to classify and localise the objects, during the training process.

The latest version of YOLOv5 (to the date of this article), incorporates 4 head outputs with the strides of 8, 16, 32 and 64. The output scale with the stride of 64 is added to improve the detection accuracy of relatively large objects in multi-faceted and comprehensive datasets. However, in most traffic monitoring scenes, including this research, the cameras are placed at a height of at least 3 metres from the ground and with a distance of more than 5 metres from the objects of interest (road users). This means the cameras hardly includes any extra-large objects that can fill up the entire image plane.

Therefore, we consider 3 different stride scales of 8, 16, 32 for the head part of our model and the k-means algorithm with 9 cluster centroids, yielding to 3 anchor boxes for each grid-cell. This means the model can detect 3 objects in each grid cell. Our preliminary evaluations confirm accuracy improvements by using the 3 head scales rather than the 4 head scales (more details in Section 4.1). The model predicts offset of bounding boxes with respect to corresponding anchor box in each grid cell. Assuming (x_c, y_c) as the top-left corner of the grid-cell, h_a and w_a as the height and width of an anchor box in that grid cell, a bounding box with the centre (x_b, y_b), height h_b and width w_b is calculated by the predicted offset (x_o, y_o, w_o, h_o) as follows:

$$\begin{aligned} x_b &= \sigma(x_o) + x_c \\ y_b &= \sigma(y_o) + y_c \\ w_b &= w_a \times e^{w_o} \\ h_b &= h_a \times e^{h_o} \end{aligned} \tag{1}$$

where σ is the Sigmoid function, normalising the input between 0 and 1.

Eventually, the model produces a set D for each image witch contains $(x_b, y_b, w_b, h_b, \mathfrak{s}, \mathbf{c})$ for each object, where $\mathfrak{s}$ is 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):

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

3.2 Object Tracking and Moving Trajectory

DeepSORT [46] is a common DNN-based object tracking algorithm that extracts appearance features to track the detected objects. However, it comes with a comparatively high computational cost which is a negative point for multi-modal applications such as our traffic surveillance application.

Therefore, we aim at enhancing the Simple Object Real-time Tracking (SORT) algorithm [47] by developing a fast tracking model called multi-object and multi-class tracker (MOMCT), while maintaining a high level of tracking accuracy.

The SORT 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 and each class needs to be dealt with separately. As a result, in some cases, the object detector assigns a new class to an object (bounding box) which is not aligned with the SORT object tracker estimation. In such cases, the tracker sees it as a new object, assigns a new ID to it and consequently loses the previous tracking.

To overcome this issue, we integrate a category vector $\acute{\mathbf{c}} \in \mathbb{W}^{1\times 11}$ for 11 categories of detected objects in the internal

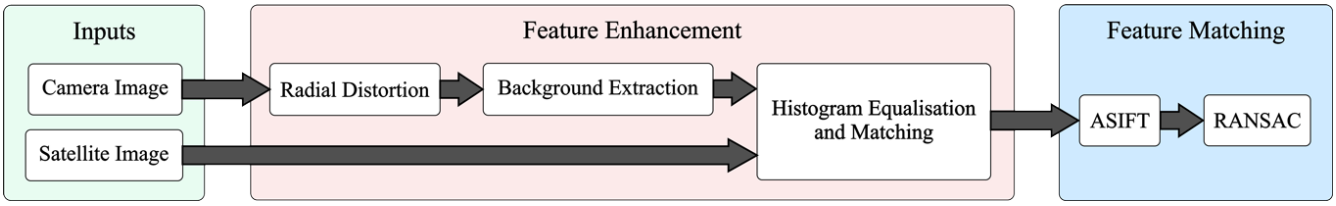


Figure 4. The hierarchical structure of the proposed feature matching model.

Kalman filter of the SORT tracker. The category vector is the one-hot encoded representation of the detected class vector $\mathbf{c}$, in which the highest class probability is shown by 1 and the rest of the probabilities by 0.

Exploiting the smoothing effect of the Kalman filter would filter out the bouncing of detected categories through the sequence of frames. Also, it enables the SORT 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 new Kalman filter is defined as follows:

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

where $s_b = w_b \times h_b$ denotes the scale (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 Kalman filter as follows:

$$\dot{\mathbf{z}} = [\ \hat{x}\ \ \hat{y}\ \ s_b\ \ r_b\ \ |\ \dot{\mathbf{c}}\ ]^T \tag{4}$$

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:

$$\begin{aligned} M_{v_i} &= \{(\hat{x}^t_{v_i}, \hat{y}^t_{v_i}) \ : \ \forall t \in T_{v_i}\} \\ M_{p_i} &= \{(\hat{x}^t_{p_i}, \hat{y}^t_{p_i}) \ : \ \forall t \in T_{p_i}\} \end{aligned} \tag{5}$$

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

$$\begin{aligned} M_V &= \{M_{v_i} \ : \ \forall v_i \in V\} \\ M_P &= \{M_{p_i} \ : \ \forall p_i \in P\} \end{aligned} \tag{6}$$

3.3 Camera Auto-calibration

The intuition behind this part of the study is to apply an automatic IPM camera calibration setup where and when no information about the camera and mounting specifications are available. This makes our study applicable for most CCTV traffic surveillance cameras in city roads and urban areas, as well as other similar applications, without the requirements of knowing the camera intrinsic parameters, height and angle of the camera.

We exploit a top-view satellite image from the same location of the CCTV camera and develop a hybrid satellite-ground based inverse perspective mapping (SG-IPM) to automatically calibrate the surveillance cameras. This is an end-to-end technique to estimate the planar transformation matrix $\mathbf{G}$ as per Equation 35 in Appendix A. The matrix $\mathbf{G}$ is used to transform the camera perspective image to a bird's eye view image.

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

Since the coordinates of the bird's eye view image approximately matches the satellite image coordinates (i.e. $\check{\mathbf{I}} \approx \grave{\mathbf{I}}$), the utilisation 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 the $\hat{\mathbf{I}}$.

In order to solve the linear equation 35, at least four pairs of corresponding points in $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ are required. Therefore, we would need to extract and match similar features pairs from both images. These feature points should be robust and invariant to rotation, translation, scale, tilt, and also partial occlusion in case of high affine variations.

Figure 4 represents the general flowchart of our SG-IPM technique, which is fully explained in the following subsections, including *feature enhancement* and *feature matching*:

3.3.1 Feature Enhancement

Three types of feature enhancement are addressed before applying the calibration processes:

- Radial distortion correction
- Background removal
- Histogram matching

Radial Distortion: Some of the road-side cameras have non-linear radial distortion due to their wide-angle lens which will affect the accuracy of the calibration process and monitoring system to estimate the location of the objects.

Such type of noise would also reduce the resemblance between $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ images, especially, in the case that we want to find similar feature points.

Examples of the barrel-shaped radial noise are shown in Figure 5, left column. Similar to a study by Dubská et al [34], we assume the vehicles traverse between the lanes in a straight line. We use the vehicles' trajectory sets to remove radial distortion noise. For each vehicle v_i, a polynomial radial distortion model is applied to the location coordinates $(\hat{x}_{v_i}, \hat{y}_{v_i})$ of the vehicle's trajectory set (M_{v_i}) as follows:

$$\begin{aligned}(\bar{x}, \bar{y}) = ((\hat{x}_{v_i} - x_s)(1 + k_1 r^2 + k_2 r^4 + ...),\\ (\hat{y}_{v_i} - y_s)(1 + k_1 r^2 + k_2 r^4 + ...))\end{aligned} \tag{7}$$

$$r = \sqrt{(\hat{x}_{v_i} - x_s)^2 + (\hat{y}_{v_i} - y_s)^2} \tag{8}$$

where $(\bar{x}, \bar{y})$ is the corrected location of the vehicle, (x_s, y_s) denotes the centre of the radial noise, $K = \{k_1, k_2, ...\}$ are the unknown scalar parameters of the model which need to be estimated, and r is the radius of the distortion with respect to the centre of the image.

A rough estimation of k_1 and k_2 would be sufficient to remove the major effects of such noise. To this regard, each point of the moving trajectories would be applied to the Equation 7 yielding to transformed trajectory set $\bar{M}_{v_i}$. Then, the optimal values of k_1 and k_2 would be achieved by minimising the sum of squared errors between the best fitting line ℓ to the M_{v_i} and $\bar{M}_{v_i}$ as follows:

$$K = \arg \min_{k} \sum_{v_i \in V} \sum_{\bar{l}_j \in \bar{M}_{v_i}} (\ell . \bar{l}_j)^2 \tag{9}$$

where $\bar{l}_j$ is the corrected pixel location of the vehicle v_i belonging to the transformed moving trajectory set $\bar{M}_{v_i}$.

Finally, the optimal parameters will be estimated using $(1+\lambda)$-ES evolutionary algorithm with $\lambda = 8$ as discussed in [34].

Background Extraction: Since $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ images are captured using two different cameras (ground camera vs. aerial satellite camera) and in different dates, times, or weather conditions, the images may seem inconsistent and different. This is mostly due to the existence of different foreground objects and road users on each image. This makes it hard to find analogous features to match.

To cope with that challenge, we extract the background of image $\hat{\mathbf{I}}$ by eliminating the moving objects. We apply an accumulative weighted sum over the intensity value for a period of n_t (frames) to remove the effect of the temporal pixel value changes as follows:

$$\hat{\mathbf{B}}^t = (1-\alpha)\hat{\mathbf{B}}^{t-1} + (\alpha\ \hat{\mathbf{I}}^t) \quad , \quad 1 \le t \le \mathsf{n}_t \tag{10}$$

Figure 5. Eliminating the radial distortion in MIO-TCD dataset samples [40]. Left column: original images. Right column: rectified images after barrel distortion removal.

where initially $\hat{\mathbf{B}}$ is the accumulative variable and $\hat{\mathbf{B}}^0$ is equal to the first input frame $\mathbf{I}^0$, α is the weighted coefficient that determines the importance of the next incoming frame. Our experiment shows that $\alpha = 0.01$, and $\mathsf{n}_t \approx 70$ frames is usually sufficient to remove the foreground objects in most urban and city roads with a moderate traffic flow.

Figure 6 shows samples of the background extraction method applied to various roads and traffic scenarios.

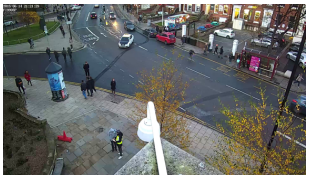
Histogram Matching: Lighting condition variation is another barrier that makes it hard to find and match similar feature points between $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$.

We utilise a colour correlation-based histogram matching [50] which adjusts the hue and luminance of $\hat{\mathbf{I}}$ and $\grave{\mathbf{I}}$ into the same range. The algorithm can be extended to find a monotonic mapping between two sets of histograms. The optimal monotonic colour mapping E is calculated to minimise the distance between the two sets simultaneously.

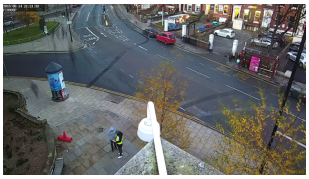
$$\vec{d} = \arg \min_{E} \sum_{i=n_p} d(E(\hat{\mathbf{I}}_i^g),\ \grave{\mathbf{I}}_i^g) \tag{11}$$

where $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$ are grey-level images, n_p is the number of pixels in each image, E is histogram matching function, and $d(\cdot,\cdot)$ is a Euclidean distance metric between two histograms.

Figures 7a and 7b show the results of the background extraction and histogram matching process, respectively.



(a) Laidlaw Library CCD original video footage



(b) The generated Laidlaw Library background video

(c) GRAM original video footage

(d) GRAM background video

(e) UA-DAT original video footage

(f) UA-DAT background video

Figure 6. Samples of background extraction on the UK Leeds Laidlaw Library (Parkinson Building) dataset, GRAM dataset [48], and UA-DAT dataset [49] .

3.3.2 Feature Matching

To handle the high affine variation between the images, we adapt the Affine Scale-Invariant Feature Transform (ASIFT) [51] method. This method generates view samples along different latitude and longitude angles of the camera. Then it applies Scale-Invariant Feature Transform (SIFT) [52] algorithm. This makes it invariant to all parameters of the affine transformation and a good candidate to match the features between $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$.

However, there might be some outliers between the matching features causing inaccurate estimation of the matrix $\mathbf{G}$. To remove these outliers, we use Random Sample Consensus (RANSAC) [53], which is an iterative learning algorithm for parameter estimation. In each iteration, the RANSAC 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 matching pairs, the following criteria can be defined to evaluate the best candidates:

$$F_n = \begin{cases} 1 & d(\Lambda(\hat{l}_f, \mathbf{G}),\ \grave{l}_f) < \tau_{\mathrm{Z}} \\ 0 & \text{Otherwise} \end{cases} \tag{12}$$

where F_n is the result of voting for the n-th pair, τ_{Z} is a distance threshold to determine whether a pair is an inlier or not, and

(a) Background extraction

(b) The outcome of histogram matching

Figure 7. Histogram matching algorithm applied to the Leeds University Laidlaw Library (Parkinson Building) surveillance camera (right column), and the satellite image of the same location (left column).

d is the Euclidean distance measure. Consequently, the total number of inlier votes ($\hbar_i$) for the matrix $\mathbf{G}$ in the i-th iteration will be calculated as follows:

$$\hbar_i = \sum_{n=1}^{\eta} F_n \quad , \quad i \in \zeta \tag{13}$$

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

$$\zeta = \frac{\log(1-\rho)}{\log(1-\epsilon^{\gamma})} \tag{14}$$

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 $\mathbf{G}$ matrix), and ρ is the probability of all ς sampled pairs being inliers in an iteration. After the end of the iterations, the $\mathbf{G}$ matrix with the highest vote will be elected as the suitable transformation matrix between $\hat{\mathbf{I}}^g$ and $\grave{\mathbf{I}}^g$.

Figure 8 represents an example of the feature matching process applied to a real-world scenario. Figures 8a and 8b show the results of the ASIFT algorithm and the RANSAC method, respectively.

Eventually, we apply the $\mathbf{G}$ matrix on coordinates of interest in $\hat{\mathbf{I}}$ (such as positions of detected objects $(\hat{x}, \hat{y})$), to estimate their corresponding coordinates in $\check{\mathbf{I}}$:

$$(\check{x}, \check{y}) = \Lambda((\hat{x}, \hat{y}), \mathbf{G}) \tag{15}$$

(a) ASIFT feature matching results

(b) RANSAC inlier matches

Figure 8. Feature matching process applied to the Leeds University Laidlaw Library (Parkinson Building) surveillance camera (right side) and the satellite image of the same location (left side).

Figure 9. Overlapping the estimated BEV image $\check{\mathbf{I}}$ to the ground truth satellite image $\grave{\mathbf{I}}$ of the same location.

As it can be visually confirmed, Figure 9 shows a very accurate result of mapping of the estimated matrix $\mathbf{G}$ on $\hat{\mathbf{I}}$ coordinates. The resulting image has been projected on the $\grave{\mathbf{I}}$ to make the intersection of overlapping areas more visible, and also easier for a visual comparison.

3.4 3D Environment Modelling and Traffic Analysis

Automated analysis of traffic scene videos via surveillance cameras is a non-trivial and 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 makes it even more complex. In this section, we elaborate our techniques of providing an abstract visual representation of the environment, objects of interest, traffic density, and traffic flow. In order to achieve a 3D bounding box modelling and representation of the road users, we require to identify and recognise the following properties for the road users and the road scene:

- Vehicle's velocity (ϑ)
- Vehicle's heading angle (θ)
- Road boundary detection

Initially, the estimation process of the vehicle's velocity (ϑ) and heading angle (θ) is described. Then we apply semantic segmentation on the satellite image to extract the road's region and boundary, and finally, we propose a method to create 3D bounding boxes.

3.4.1 Speed Estimation

Assuming the location of vehicle v_i in the current time as $\check{l}^t_{v_i} = (\check{x}^t_{v_i}, \check{y}^t_{v_i})$ and in the previous time 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 can be calculated as follows:

$$\vartheta_{v_i} = \frac{d(\check{l}^t_{v_i}, \check{l}^{t-1}_{v_i})}{\Delta t} \times \iota \tag{16}$$

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

Although the integrated Kalman filter of the object tracker in the perspective image reduces the object localisation noise to some extent, the SG-IPM method may add up some additional noise in the bird's eye view image, which in return leads to an unstable bird's eye view mapping and estimations. To overcome this issue, we have applied a constant acceleration Kalman filter on the object locations $(\check{x}, \check{y})$ which models the motion of objects. The state matrix of this Kalman filter is defined as:

$$\check{\mathbf{x}} = [\ \check{x}\ \ \check{y}\ \ \dot{x}\ \ \dot{y}\ \ \ddot{x}\ \ \ddot{y}\ ]^T \tag{17}$$

where the $\dot{x}$ and $\dot{y}$ are the velocity and $\ddot{x}$ and $\ddot{y}$ are the accelerations in $\check{x}$ and $\check{y}$ directions, respectively.

We represent the Kalman transition matrix ($\check{\mathbf{A}}$) as follows:

$$\check{\mathbf{A}} = \begin{bmatrix} 1 & 0 & t_w & 0 & \frac{t_w^2}{2} & 0 \\ 0 & 1 & 0 & t_w & 0 & \frac{t_w^2}{2} \\ 0 & 0 & 1 & 0 & t_w & 0 \\ 0 & 0 & 0 & 1 & 0 & t_w \end{bmatrix} \tag{18}$$

where $t_w = \frac{1}{fps}$ is the real-world time between the former and the current frame depending on the camera frame rate (frame per second, *fps*). The observation matrix $\check{\mathbf{z}}$ can be defined as follows:

$$\check{\mathbf{z}} = [\ \check{x}\ \ \check{y}\ ]^T \tag{19}$$

Using the Kalman-based smoothed location $(\check{x}, \check{y})$ and the frame by frame velocity of objects $\dot{x}$, the speed of a vehicle will be calculated (in *mph*) as follows:

$$\vartheta_{v_i} = \dot{x}_{v_i} \times \iota \tag{20}$$

where $\dot{x}_{v_i}$ is the "pixels per second" velocity of the vehicle v_i,, and ι is the pixel-to-mile ratio. Samples of estimated speeds (in mph) is shown on top-left corner of the vehicle bounding boxes in Figure 1, bottom row.

In case of missing observations due to e.g. partial occlusion, we predict the current location of vehicles using the process step of the Kalman filter ($\check{\mathbf{A}}$.$\check{\mathbf{x}}$) and buffering the predicted locations up to an arbitrary number of frames.

3.4.2 Angle Estimation

The heading angle of a vehicle can be calculated as follows:

$$\theta_{v_i} = \theta(\check{l}^t_{v_i}, \check{l}^{t-1}_{v_i}) = \tan^{-1}\left(\frac{\check{y}^t_{v_i} - \check{y}^{t-1}_{v_i}}{\check{x}^t_{v_i} - \check{x}^{t-1}_{v_i}}\right) \tag{21}$$

The angle estimation is very sensitive to the displacement of vehicle locations, and even a small noise in localisation can lead to a significant change in the heading angle. However, in the real world the heading angle of vehicles would not change significantly in a very short period of time (e.g. between two consequent frames).

We introduce a simple yet efficient Angle Bounce Filtering (ABF) method to restrict sudden erroneous angle changes between the current and previous angle of the vehicle:

$$\Delta\theta_{v_i} = \theta^t_{v_i} - \theta^{t-1}_{v_i} \tag{22}$$

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:

$$\mathsf{w} = \frac{\cos((4\pi \times \tilde{\Delta}) + 1)}{2} \tag{23}$$

where $\tilde{\Delta}$ is the normalised 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 neutralise 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. Figure 10 illustrates the smoothed values of w by green colour spectrum. The darker green, the lower the coefficient.

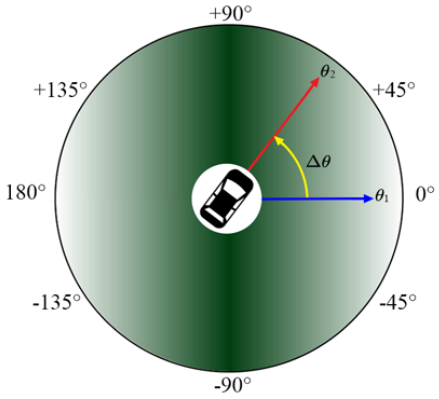


Figure 10. $\Delta\theta$ cosine suppression operation. The darker zones receive a lower coefficients which in turn suppress any large and sudden angular changes between two consequent frames.

Finally, we rectify the vehicle-angle as follows:

$$\tilde{\theta}^t_{v_i} = \theta^{t-1}_{v_i} + (\mathsf{w} \times \Delta\theta_{v_i}) \tag{24}$$

In some cases the moving trajectory may not be available; for instance, when a vehicle appear on the road-scene for the first time or some vehicles are stationary (or parked) during their entire presence in the scene. For such cases the heading direction of the vehicle cannot be directly estimated as no prior data is available about the vehicle movement history. However, we can still calculate the angle of the vehicles by calculating a perpendicular line from the vehicle position to the closest boundary of the road. Identifying the border of the road requires a further road segmentation operation.

Some of the existing deep-learning based studies such as [54] mainly focus on segmenting satellite imagery which are captured from a very high altitude and heights comparing to the height of CCTV surveillance cameras.

Moreover, there are no annotated data available for such heights to train a deep-learning based road segmentation model. In order to cope with that limitation, we adapt a Seeded Region Growing method (SRG) [55] on intensity values of the image $\check{\mathbf{I}}$.

We consider the moving trajectory of vehicles traversing the road in $\check{\mathbf{I}}$ domain ($\Lambda(M_{v_i}, \mathbf{G})\ \ \forall v_i \in V$), as initial seeds for the SRG algorithm. In the first step, the algorithm calculates the intensity difference between each seed and its adjacent pixels. Next, the pixels with an intensity distance less than a threshold τ_α, are considered as connected regions to the seeds. Utilising these pixels as new seeds, the algorithm repeats the above steps until no more connected regions are found. At the end of the iterations, the connected regions represent the segment of the road.

Figure 11. Road segmentation on the satellite image. The red lines represent the initial segmentation result extracted from the SRG method, and the green region is the final segmentation output after applying the morphological operations.

Figure 12. Reference angle estimation process with respect to nearest road boundary. The boundaries are denoted with green lines, which are extracted by application of the Canny edge detector on the road segment borders.

Due to a large intensity variations among adjacent pixels in the road segment (such as white lane markings vs the dark grey asphalt coatings), there might be some fragmented road boundary segments as shown in Figure 11 (the regions denoted by red lines at the centre and around the road).

We apply morphological dilation operations with 3×3 kernel size, to expand the segmented area and fill these small gap regions. Also, an erosion operation with the same kernel size is performed to smooth the road region by removing the sharp edges and spikes of the road boundaries. Figure 11, green regions, represent the segmentation results.

The road segmentation process can be done as an offline procedure before the real-time traffic monitoring operation starts. The scene need to be monitored until sufficient vehicle locations (seeds) are detected to segment the entire road region. Since the initial seeds are moving trajectories of vehicles, the monitoring time may vary for different scenes depending on the presence of the vehicles traversing the road. Based on our experience this may vary from 5 seconds to 5 minutes depending on the live traffic flow.

In order to calculate the reference heading angle for each vehicle (v_i), we find a point ($l^r_{v_i}$) on the road border which has the minimum Euclidean distance to the vehicle's central location. This distance is shown by the red dash-line in Figure 12 which is perpendicular to the road border.

We consider a small circle (the blue circle) with negligible radius $\mathfrak{r}$ centring at $l^r_{v_i}$. Then, we find locations of two points (Ψ_1 and Ψ_2), in which the circle intersects the road boundary. Finally, similar to a derivative operation, the heading angle is calculated by $\theta(\Psi_1, \Psi_2)$, which represents the slope of the red lines at the road boundary, as well as the vehicle heading angle (Figure 12).

3.4.3 2D to 3D Bounding Box Conversion

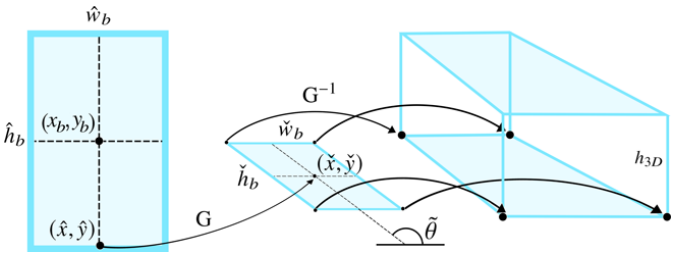
In order to determine the occupied space of each object in $\hat{\mathbf{I}}$ domain, we convert a 2D bounding box (Figure 13a) to a cubical 3D bounding box by estimating 8 cube's corners. The cube's floor consists of 4 corner points and corresponds to a rectangle in the $\check{\mathbf{I}}$ domain (Figure 13b the middle shape). This rectangle indicates the area of the ground plate which is occupied by the object, and can be addressed with the centre $(\check{x}, \check{y})$, the height $\check{h}_b$ and the width $\check{w}_b$. The $\check{h}_b$ and $\check{w}_b$ are determined based on prior knowledge about the approximate height and width of the corresponding object's category in real world (i.e. 5.80×2.9 meter for buses in the UK). In order to have these distances in pixel criterion, we divide them by the pixel-to-meter ratio (ι), as explained in the Speed Estimation section (3.4.1).

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. Afterwards, we add h_{3D} to the y axis of the floor corners to indicate the 4 points of the cube's roof.

The height of the cube for all road users, except the pedestrians, is set $h_{3D} = \beta \times h_b$, where $\beta = 0.6$ is determined by our experiments as a suitable height coefficient for the detected bounding boxesin 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 ($h_{3D} = h_b$).

Figure 13a, 13b, 13c show the hierarchical steps of our approach from 2D to 3D conversion on Leeds University

(a) Detected objects in 2D bounding boxes



(b) 2D to 3D bounding box conversion

(c) Final 3D representation

Figure 13. 2D to 3D bounding box conversion process in four categories of vehicle/truck, pedestrian, bus, and cyclist.

Laidlaw Library surveillance camera footage.

4 Experiments

In this section, we evaluate the performance and accuracy of the proposed 3D road-users detection model followed by assessing the efficiency of the proposed environment modelling.

4.1 Performance Evaluation

The majority of modern object detectors are trained and evaluated on large and common datasets such as Microsoft Common Objects in Context (Ms-COCO) [39]. The COCO dataset consists of 886,284 samples of general annotated objects for 80 categories (include person, animal, appliance, vehicle, accessory etc.) in 123,287 images. However, none of them is dedicated to traffic monitoring purposes.

We considered the MIO-TCD dataset [40] which consists of 648,959 images and 11 traffic-related annotated categories (including cars, pedestrian, bicycle, bus, three types of trucks, two types of vans, motorised vehicles, and non-motorised vehicles) to train and evaluate our models. The dataset has been 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 1, we also considered two more traffic monitoring video-footage including UA-DETRAC [49] and GRAM Road-Traffic Monitoring (GRAM-RTM) [48] to test our models under different weather and day/night lighting conditions. Moreover, we set up our own surveillance camera at one of the highly interactive intersections of Leeds City, near the Parkinson Building at the University of Leeds, to further evaluate the performance of our model on a real-world scenario consisting of 940,000 video frames from the live traffic.

As mentioned in the Methodology section (3.1), we adopted transfer learning to train different architectures of YOLOv5 model on the MIO-TCD dataset. We exploited pre-trained weights of 80 class COCO dataset as initial weights of our fine-tuning process.

There are four versions of YOLOv5 which are distinguished by the number of learning parameters. The “small” with 7.5 million parameters is a lightweight version, “medium” version (21.8 million), “large” (47.8 million), and “xlarg” version which has 89 million learnable parameters. We performed experiments with different number of head modules which consist of three or four head outputs to classify different sizes of objects (as described in section 3.1).

In the training phase (Figure 14a), we minimised the loss function of the adapted YOLOv5, 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 optimal learning-rate and avoid long training time, we used one-cycle-learning-rate [56]. This gradually increases the learning rate to a certain value (called 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.

Figure 14 illustrates the analytic graphs of the training and validation processes. As per the classification graphs (Fig. 14a), the training loss starts decreasing around epoch 35, while the validation loss starts increasing (Fig. 14b). This is a sign of over-fitting in which the model starts memorising the dataset instead of learning generalised features. To avoid the effects of over-fitting, we choose the optimal weights which yield the minimum validation loss.

Table 2 compares the performance of the proposed YOLOv5-based model with 10 other state-of-the-art object detection method on the challenging dataset of MIO-TCD. Two metrics of *mAP* and speed (*fps*) are investigated.

Table 1. Specifications of the test datasets used in this research, including various weather conditions, resolutions, frame rates, and video lengths.

Dataset	Weather	Length (frame)	Resolution	*fps*
UA-DET [49]	Sunny, Rainy, Cloudy, Night	140000	960×540	25
GRAM-RTM [48]	Sunny, Foggy	40345	1200×720	30
UK Leeds Parkinson	Day, Sunset, Night	940000	1920×1080	30

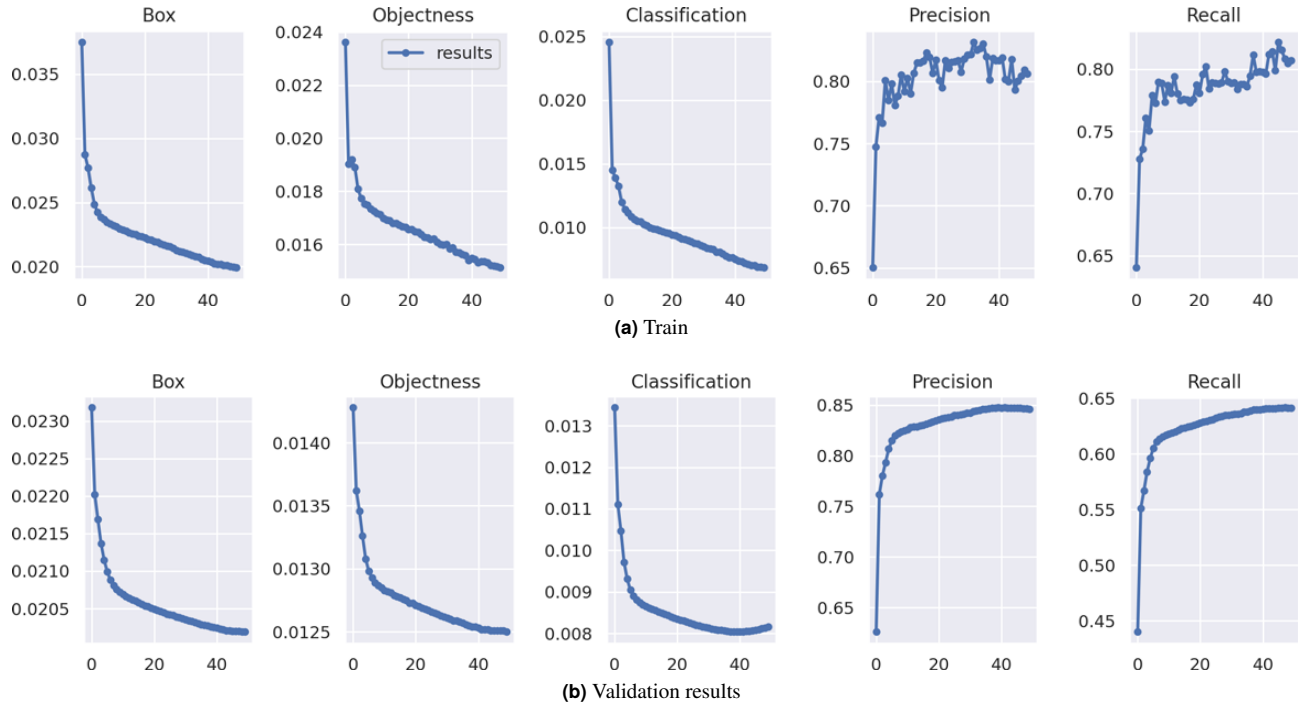


Figure 14. Error minimisation graphs of the model in training and validations phases, after 50 epochs.

As can be seen, the adapted YOLOv5-based model has achieved a considerable increase in mean average precision comparing to the former standard YOLOv4 algorithm (84.6% versus 80.4%). The experiments also proved that 3-head versions of YOLOv5 provides more efficiency in traffic monitoring than the 4-head versions. The lightweight version of YOLOv5 reaches the highest rate of speed ($123 fps$). While the model has sacrificed the accuracy by -1.7% in comparison to the highest rate (84.6%).

The YOLOv5 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. Whereas the 3-head YOLOv5 Large, has the same mAP score, and provides a real-time performance of $36.5 fps$. This makes the model more suitable for the cases in which heavy post-processing procedures are involved.

Table 3 shows the test results of our pioneer detection model (YOLOv5-Large, 3 head) on UA-DET and GRAM-RTM datasets with very high precision rates of 99.8% and 99.7%, respectively. The GRAM-RTM dataset only provides ground truth annotations for one lane of the road. So, we applied a mask to ignore the non-annotated lanes of the road; otherwise, our proposed model is capable of detecting vehicles in both lanes.

Figure 15, top row, shows the results of the detection algorithm and 3D localisation of the road users. Figure 15, the bottom row, shows the environment modelling of the scene as a digital twin of the scene and live traffic information. Such live information (which can be stored in cloud servers), would be be very useful for city councils, police, governmental authorities, traffic policy makers, and even as extra source of processed data for automated vehicles (AVs) which traverse around the same zone. Such rich digital twins of the road condition can significantly along with the ego-vehicles sensory data can enhance the AVs' capability in better dealing with the corner cases and complicated traffic scenarios.

In Figure 15 we are also trying to show the efficiency of the heading angle estimation and the tracking system in case of

Table 2. A comparison of mean average precision (mAP) rate between the developed models and 10 other models on MIO-TCD dataset. The accuracy scores of 3 truck categories (Articulate Truck, Pickup Truck and Single Unit Truck) is averaged and presented in a one column- "Trucks × 3".

Method	Speed (fps)	mAP	Bicycle	Bus	Car	Motorcycle	Motorised Vehicle	Non-motorised Vehicle	Pedestrian	Work Van	Trucks × 3
Faster-RCNN [40]	9	70.0 %	78.3%	95.2%	82.6%	81.1%	52.8%	37.4%	31.3%	73.6%	79.2%
RFCN-ResNet-Ensemble4 [57]	-	79.2%	87.3%	97.5%	89.7%	88.2%	**62.3%**	59.1 %	48.6 %	79.9 %	86.4%
SSD-512 [40]	16	77.3%	78.6%	96.8%	94.0%	82.3%	56.8%	58.8%	43.6%	80.4%	86.4 %
Context ModelA [58]	-	77.2%	79.9%	96.8%	93.8%	83.6%	56.4%	58.2%	42.6%	79.6%	86.1%
Adaptive Ensemble [59]	-	74.2%	82.2%	95.7%	91.8%	87.3%	60.7%	45.7 %	47.9%	63.8%	80.5%
SSD-300 [40]	16	74.0%	78.3%	95.7%	91.5%	78.9%	51.4%	55.2%	37.3%	75.0%	83.5%
YOLOv2-MIOTCD [40]	18	71.8%	78.6%	95.1%	81.4%	81.4%	51.7%	56.6%	25.0%	76.4%	81.3%
YOLOv2-PascalVOC [40]	18	71.5%	78.4%	95.2%	80.5%	80.9%	52.0%	56.5%	25.7 %	75.7%	80.4 %
YOLOv1 [40]	19	62.7 %	70.0%	91.6%	77.2%	71.4%	44.4%	20.7%	18.1%	69.3%	75.5%
Our Experiments											
YOLOv4	24	80.4%	89.2%	95.8%	91.6%	91.5%	58.6%	63.9%	63.4%	79.0%	83.7 %
YOLOv5 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 %
YOLOv5 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 %
YOLOv5 Large (3 head)	36.50	84.6%	92.5%	98.7%	95.9%	94.3%	51.7%	70.1%	77.4%	83.8%	88.8%
YOLOv5 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%
YOLOv5 Large (4 head)	117.6	80.9%	91.2%	97.8%	95.1%	91.7%	48.4%	61.9%	64.3%	80.0%	86.6%
YOLOv5 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%
YOLOv5 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%
YOLOv5 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%

Table 3. Detection performance of our YOLOv5 Large (3 head) model on two auxiliary traffic-related datasets.

Datasets	Precision	Recall
UA-DET [49]	99.8%	99.7%
GRAM-RTM [48]	99.7%	99.5%

full occlusions. As can be seen, one of the cars in the scene is taking a U-turn and we have properly identified the heading angle of the car at frame 82100 (indicated with blur arrow). This can be compared with its previous angle and position in frame 82000. Considering the position and the heading angle of the vehicle at frames 82000 and 82100, the 3D bounding box of the vehicle is also determined.

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

4.2 Environment Modelling and Traffic Analysis

In order to take the most of the detection and tracking algorithms and to provide smart traffic monitoring analysis, we defined three possible states for vehicles and pedestrians as follows:

- **Parking**: a set $\mathcal{P}$ contains all of the vehicles which have less than one-meter distance in $\check{\mathbf{I}}$ domain from the road border ($l^r_{v_i}$), and their temporal speeds (ϑ_{v_i}) have been close to zero for more than 1 minute.
- **Speeding Violation**: a set $\mathcal{S}$ consists of vehicles in which their speed (ϑ_{v_i}) is more than the speed limit of the road (i.e. 30 *mph* for Leeds city centre, UK).
- **Collision Risk**: a set $\mathcal{D}$ consists of pedestrians whose distances from vehicles are less than a meter, and the vehicles are not in the parking status $\mathcal{P}$.

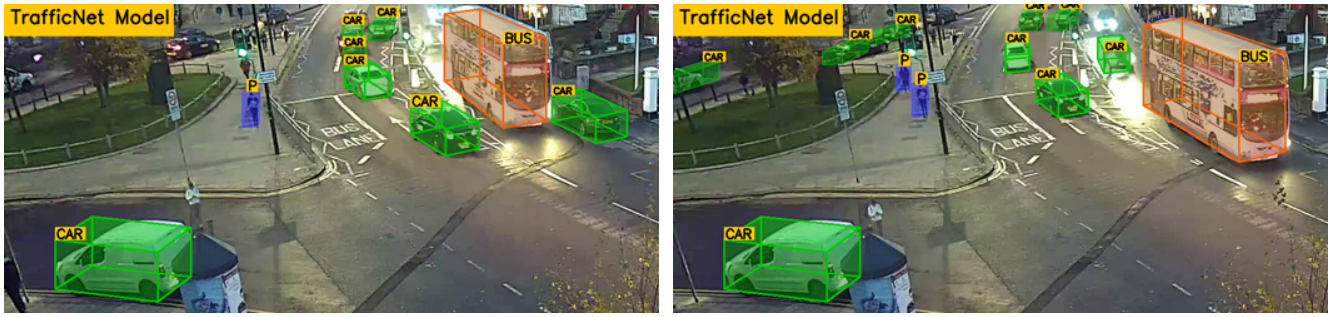
To analyse the traffic condition, we buffer the count of tracked vehicles and pedestrians locations during a period of time (e.g. 6,000 frames) as shown by line graph in Figure 16a.

In order to visualise a long-term spatio-temporal statistical analysis of traffic flow and interactions between road users, a heat map representation is created similar to our previous work in another context for social distancing monitoring [60]. The heat map 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 heat map updates in each frame by the function $G_{(\text{object})}(\check{\mathbf{H}})$ which applies a 3×3 Gaussian matrix centred at the object's location $(\check{x},\check{y})$ on the $\check{\mathbf{H}}$ matrix. Finally, we normalise the heat map intensity values between 0 and 255, in order to visualise it as a colour-coded heat image. Then a colour spectrum will be mapped to the stored values in the $\check{\mathbf{H}}$ matrix in which the red-spectrum represents the higher values, and the blue-spectrum represents the low values.

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

$$\check{\mathbf{H}}^t_{(p)} = G_{(p_i)}(\check{\mathbf{H}}^{t-1}_{(p)}) \qquad \forall p_i \in P \tag{25}$$

Figure 16b illustrates the developed heat map $\check{\mathbf{H}}_{(p)}$on the satellite image. The lower range values have been removed for



(a) Detected cars and bus on frame 82000

(b) Detected cars and bus on frame 82100

(c) Environment mapping result of frame 82000

(d) Environment mapping result of frame 82100

Figure 15. The outputs of adapted 3-head YOLOv5-large algorithm for road-user detection and environment modelling.

better visualisation. This figure provides valuable information about the pedestrians' activity. For instance, we can see a significant number of pedestrians have crossed the dedicated zebra-crossing shown by the green rectangle. However, in another region of the road (marked by a red rectangle) many other pedestrians cross another part of the road where there is no zebra-crossing. Also, there are a few pedestrians who has crossed the street directly in front of the bus station.

Similarly, the heat map for detected vehicles is defined as follows:

$$\check{\mathbf{H}}^{t}_{(v)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(v)}) \qquad \forall v_i \in V \ , \ v_i \notin \mathcal{P} \tag{26}$$

where $\check{\mathbf{H}}_{(v)}$ stores the location of moving vehicles only (not stationary or parked vehicles). This matrix has illustrated in Figure16c. This heat map represents that more vehicles are traversing on the left lane of the road comparing to the opposite direction, on the right lane.

The heat map images can be also mapped to the perspective space by: $\hat{\mathbf{H}} = \Lambda(\check{\mathbf{H}}, \mathbf{G}^{-1})$. Figures 16d and 16e are corresponded maps of Figures 16b and 16c, respectively.

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

$$\check{\mathbf{H}}^{t}_{(\vartheta)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(\vartheta)}) \qquad \forall v_i \in \mathcal{S} \tag{27}$$

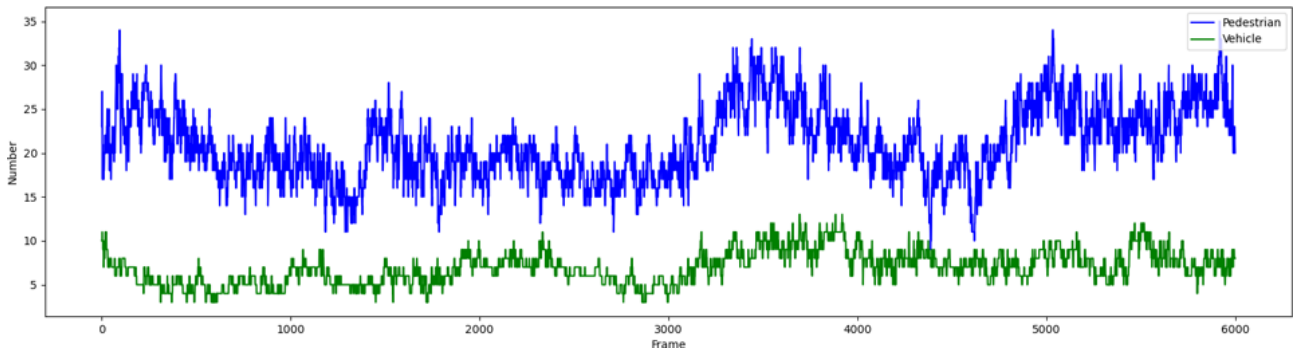
Figure 17a and 17b, illustrates an instance of speed heat map calculated over the 10,000 selected frames. As can be seen the speeding violation significantly decreases near the pedestrian crossing zone, which makes sense. As a very useful application of our developed model, similar investigations can 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 Figure 17c, 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:

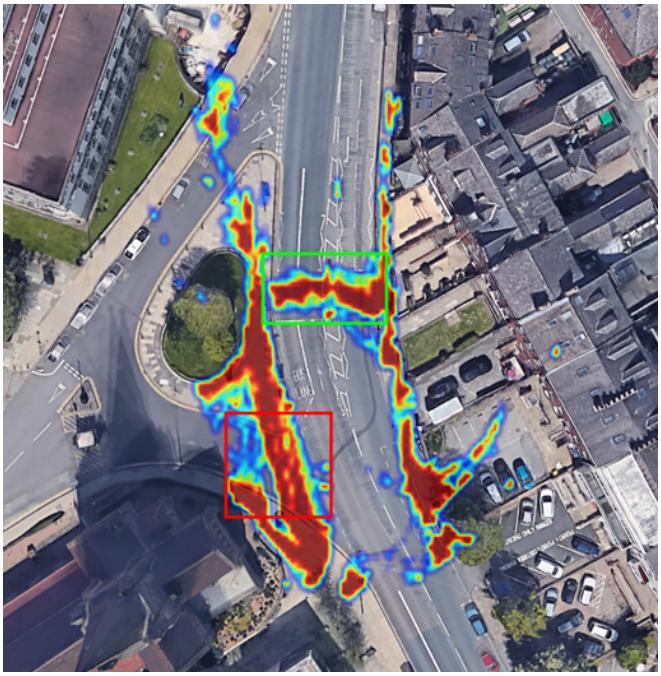
$$\bar{\vartheta} = \frac{\sum \vartheta_{v_i}}{n_v} \qquad \forall v_i \in V, v_i \notin \mathcal{P} \tag{28}$$

where n_v is the number of vehicles that are not in the Parking state.

In order to identify the congested and crowded spots in the scene, we can monitor the vehicles e.g. with less than $2m$ distances to each other with an average speed of e.g. lower than $5mph$. The shorter vehicles' proximity over a longer period of time, the larger values will be stored in the congestion buffer; consequently, a hotter heat map will be generated. Defining optimum values of distance and speed threshold requires and intensive analytical and statistical data collection

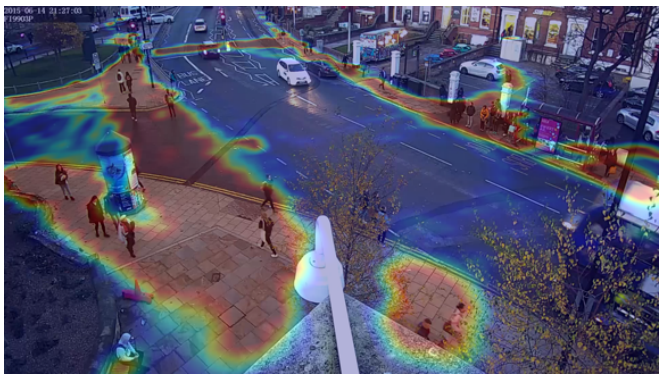


(a) Vehicle and pedestrian counts over 6000 video frames. Source: Parkinson building CCTV surveillance camera.



(b) BEV Pedestrian movements heat map

(c) BEV vehicle movements heat map



(d) Pedestrian movements heat map- Perspective view

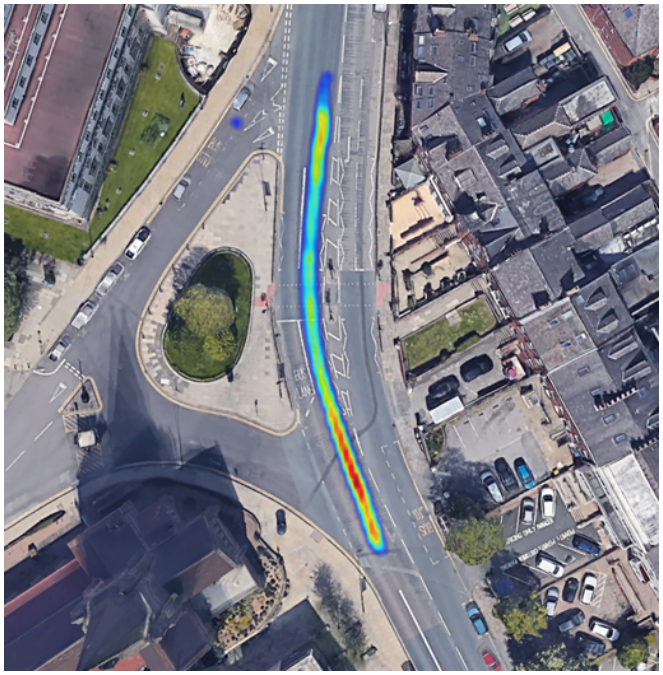
(e) Vehicle movements heat map- Perspective view

Figure 16. Spatio-temporal long-term analysis of vehicles and pedestrians' activity using Parkinson Building surveillance camera, Leeds, UK

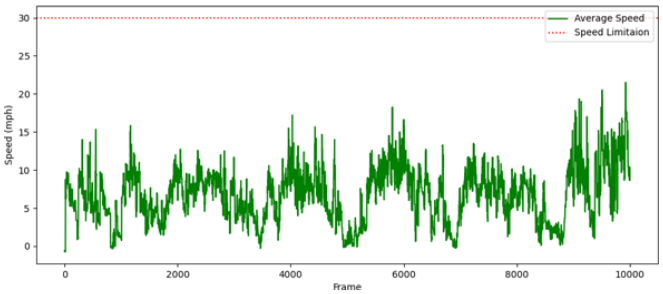
and assessments based on the road type (e.g. highway or a city road) which is out of the scope of this research.

However, as a general-purpose solution and similar to the previous heat maps, we defined the congestion heat map $\check{\mathbf{H}}_{(C)}$

(a) Speed violation heat map- Perspective view



(b) BEV speed violation heat map



(c) Average speed of moving vehicles in the scene

Figure 17. Automated speed monitoring and heat map analysis based on 10,000 video frames from the Laidlaw Library surveillance camera, Leeds, UK

as follows:

$$\check{\mathbf{H}}^{t}_{(C)} = G_{(v_i)}(\check{\mathbf{H}}^{t-1}_{(C)}) \qquad \forall v_i \in \mathcal{A} \tag{29}$$

where $\mathcal{A}$ is an ID set of vehicles that are in the congested areas.

Figure 18. Heat map representation of congested areas based on 10,000 live video frames from Woodhouse lane, Leeds LS2 9JT, UK.

Figure 19. Heat map representation of areas in which vehicles and pedestrians were too close to each other. Source/Location: 10,000 live video frames, Woodhouse lane, Leeds LS2 9JT, UK.

As we can see in Figure 18, there are two regions of congestion, one before the 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.

Figure 19 shows the pedestrian behaviour's heat map by monitoring the pedestrians who are not maintaining a minimum safety distance of $2m$ to the passing vehicles. Similarly, the heat map of the high-risk pedestrians can be updated according to the following equation:

$$\check{\mathbf{H}}^{t}_{(\mathcal{W})} = G_{(p_i)}(\check{\mathbf{H}}^{t-1}_{(\mathcal{W})}) \qquad \forall p_i \in \mathcal{D} \tag{30}$$

The hot area in front of the bus station is more likely caused by the buses which stop just beside the bus station. The heat map 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 and performance evaluations in this research were conducted on a PC workstation with an Intel ©Core™ i5-9400F processor and an NVIDIA RTX 2080 GPU with CUDA version 11. All services were performed based on a unified software using parallel processing for simultaneous utilisation 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 services 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

In this article, we proposed a real-time traffic monitoring system called Traffic-Net which applies a customised 3-head YOLOv5 model to detect various categories of vehicles and pedestrian. A multi-class and multi-object tracker named MOMCT were also developed for an accurate and continuous classification, identifications, and localisation of the same objects over consequent video frames, as well as prediction of the next position of vehicles in case of missing information. In order to develop a general-purpose solution applicable on the majority of traffic surveillance cameras, we introduced an automatic camera calibration techniques (called SG-IPM) to estimate real-world positions and distances using a combination of near-perpendicular satellite images and ground information.

Having the real-world position of the vehicles, a constant acceleration Kalman filter was applied for smooth speed estimation. Using spatio-temporal moving trajectory information, the heading angle of vehicles were also calculated. We also introduced the ABF method to remove the angle variation noise due to occlusion, sensor limitation, or detection imperfection.

These led to 3D bounding box estimation and traffic heat map modelling and analysis which can help the researchers and authorities to automatically analyse the road congestion, high-risk areas, and the pedestrian-vehicle interactions. Experimental results on the MIO-TCD dataset and a real-world road-side camera, confirmed the proposed approach well dominates 10 state-of-the-art research work in ten categories of vehicles and pedestrian detection. 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 outcomes of this research.

As a future study and in order to improve the feature matching process between the camera and satellite images, a neural network-based feature matching algorithm can be applied to increase the accuracy. Also, many other strategies (like evolutionary algorithms, feature engineering, and generative models) can be used to provide more robust features, to tackle the matching failures.

Availability of larger datasets can further help to improve the accuracy of heat maps, to identify high-risk road spots and further statistical analyses.

Acknowledgement

The research has received funding from the European Commission Horizon 2020 program under the L3Pilot project, grant No. 723051 as well as the interACT project from the European Union's Horizon 2020 research and innovation program, grant agreement No. 723395. Responsibility for the information and views set out in this publication lies entirely with the authors.

References

1. Nambiar, R., Shroff, R. & Handy, S. Smart cities: Challenges and opportunities. In *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, 243–250, DOI: 10.1109/COMSNETS.2018.8328204 (2018).

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

3. Olatunji, I. E. & Cheng, C.-H. Video analytics for visual surveillance and applications: An overview and survey. *Mach. Learn. Paradigms* 475–515, DOI: 10.1007/978-3-030-15628-2_15 (2019).

4. Mondal, A., Dutta, A., Dey, N. & Sen, S. Visual traffic surveillance: A concise survey. In *Frontiers in Artificial Intelligence and Applications*, vol. 323, 32–41, DOI: 10.3233/FAIA200043 (IOS Press, 2020).

5. Poddar, M., Giridhar, M., Prabhu, A. S., Umadevi, V. *et al.* Automated traffic monitoring system using computer vision. In *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, 1–5, DOI: 10.1109/ICTBIG.2016.7892717 (IEEE, 2016).

6. Hu, W., Tan, T., Wang, L. & Maybank, S. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Syst. Man Cybern. Part C: Appl. Rev.* **34**, 334–352, DOI: 10.1109/TSMCC.2004.829274 (2004).

7. Yang, W., Fang, B. & Tang, Y. Y. Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road. *IEEE Transactions on Syst. Man, Cybern. Syst.* **48**, 755–766, DOI: 10.1109/TSMC.2016.2616490 (2018).

8. Oliveira, M., Santos, V. & Sappa, A. D. Multimodal inverse perspective mapping. *Inf. Fusion* **24**, 108–121, DOI: https://doi.org/10.1016/j.inffus.2014.09.003 (2015).

9. Brunetti, A., Buongiorno, D., Trotta, G. F. & Bevilacqua, V. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing* **300**, 17–33, DOI: 10.1016/j.neucom.2018.01.092 (2018).

10. Rezaei, M., Terauchi, M. & Klette, R. Robust vehicle detection and distance estimation under challenging lighting

conditions. *IEEE Transactions on Intell. Transp. Syst.* **16**, 2723–2743, DOI: 10.1109/TITS.2015.2421482 (2015).

11. Gawande, U., Hajari, K. & Golhar, Y. Pedestrian detection and tracking in video surveillance system: Issues, comprehensive review, and challenges. *Recent Trends Comput. Intell.* DOI: 10.5772/intechopen.90810 (2020).

12. Cheung, S.-c. S. & Kamath, C. Robust techniques for background subtraction in urban traffic video. In Panchanathan, S. & Vasudev, B. (eds.) *Visual Communications and Image Processing 2004*, vol. 5308, 881–892, DOI: 10.1117/12.526886. International Society for Optics and Photonics (SPIE, 2004).

13. Zhou, J., Gao, D. & Zhang, D. Moving vehicle detection for automatic traffic monitoring. *IEEE Transactions on Veh. Technol.* **56**, 51–59, DOI: 10.1109/TVT.2006.883735 (2007).

14. Chintalacheruvu, N., Muthukumar, V. *et al.* Video based vehicle detection and its application in intelligent transportation systems. *J. transportation technologies* **2**, 305, DOI: 10.4236/jtts.2012.24033 (2012).

15. Cheon, M., Lee, W., Yoon, C. & Park, M. Vision-based vehicle detection system with consideration of the detecting location. *IEEE Transactions on Intell. Transp. Syst.* **13**, 1243–1252, DOI: 10.1109/TITS.2012.2188630 (2012).

16. Zou, Z., Shi, Z., Guo, Y. & Ye, J. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055* (2019).

17. Jiao, L. *et al.* A survey of deep learning-based object detection. *IEEE Access* **7**, 128837–128868, DOI: 10.1109/ACCESS.2019.2939201 (2019).

18. Liu, W. *et al.* Ssd: Single shot multibox detector. In *European conference on computer vision*, 21–37, DOI: 10.1007/978-3-319-46448-0_2 (Springer, 2016).

19. Arinaldi, A., Pradana, J. A. & Gurusinga, A. A. Detection and classification of vehicles for traffic video analytics. *Procedia Comput. Sci.* **144**, 259–268, DOI: https://doi.org/10.1016/j.procs.2018.10.527 (2018). INNS Conference on Big Data and Deep Learning.

20. Peppa, M. V. *et al.* Towards an end-to-end framework of cctv-based urban traffic volume detection and prediction. *Sensors* **21**, DOI: 10.3390/s21020629 (2021).

21. Bui, K.-H. N., Yi, H. & Cho, J. A multi-class multi-movement vehicle counting framework for traffic analysis in complex areas using cctv systems. *Energies* **13**, DOI: 10.3390/en13082036 (2020).

22. Mandal, V., Mussah, A. R., Jin, P. & Adu-Gyamfi, Y. Artificial intelligence-enabled traffic monitoring system. *Sustainability* **12**, 9177, DOI: 10.3390/su12219177 (2020).

23. Arnold, E. *et al.* A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intell. Transp. Syst.* **20**, 3782–3795, DOI: 10.1109/tits.2019.2892405 (2019).

24. Zhang, Z., Zheng, J., Xu, H. & Wang, X. Vehicle Detection and Tracking in Complex Traffic Circumstances with Roadside LiDAR. *Transp. Res. Rec.* **2673**, 62–71, DOI: 10.1177/0361198119844457 (2019).

25. Zhang, J., Xiao, W., Coifman, B. & Mills, J. P. Vehicle Tracking and Speed Estimation From Roadside Lidar. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **13**, 5597–5608, DOI: 10.1109/JSTARS.2020.3024921 (2020).

26. Song, Y., Yao, J., Ju, Y., Jiang, Y. & Du, K. Automatic detection and classification of road, car, and pedestrian using binocular cameras in traffic scenes with a common framework. *Complexity* **2020**, DOI: 10.1155/2020/2435793 (2020).

27. Alldieck, T., Bahnsen, C. H. & Moeslund, T. B. Context-aware fusion of rgb and thermal imagery for traffic monitoring. *Sensors* **16**, DOI: 10.3390/s16111947 (2016).

28. Fernandes, D. *et al.* Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy. *Inf. Fusion* **68**, 161–191, DOI: 10.1016/j.inffus.2020.11.002 (2021).

29. Zhou, T., Fan, D.-P., Cheng, M.-M., Shen, J. & Shao, L. Rgb-d salient object detection: A survey. *Comput. Vis. Media* 1–33, DOI: 10.1007/s41095-020-0199-z (2021).

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

31. Xie, J., Girshick, R. & Farhadi, A. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, 842–857, DOI: 10.1007/978-3-319-46493-0_51 (Springer, 2016).

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

33. Rezaei, M. & Klette, R. Computer vision for driver assistance. *Cham: Springer Int. Publ.* **45**, DOI: https://doi.org/10.1007/978-3-319-50551-0 (2017).

34. Dubská, M., Herout, A., Juránek, R. & Sochor, J. Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intell. Transp. Syst.* **16**, 1162–1171, DOI: 10.1109/TITS.2014.2352854 (2015).

35. Sochor, J., Juránek, R. & Herout, A. Traffic surveillance camera calibration by 3D model bounding box alignment for accurate vehicle speed measurement. *Comput. Vis. Image Underst.* **161**, 87–98, DOI: https://doi.org/10.1016/j.cviu.2017.05.015 (2017).

36. Song, H. *et al.* 3d vehicle model-based ptz camera auto-calibration for smart global village. *Sustain. Cities Soc.* **46**, 101401, DOI: https://doi.org/10.1016/j.scs.2018.12.029 (2019).

37. Kim, Z. Camera calibration from orthogonally projected coordinates with noisy-ransac. In *2009 Workshop on*

Applications of Computer Vision (WACV), 1–7, DOI: 10.1109/WACV.2009.5403107 (2009).

38. Jocher, G. *et al.* ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations, DOI: 10.5281/zenodo.4679653 (2021).

39. Lin, T.-Y. *et al.* Microsoft coco: Common objects in context (2015). 1405.0312.

40. Luo, Z. *et al.* Mio-tcd: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Process.* **27**, 5129–5141, DOI: 10.1109/TIP.2018.2848705 (2018).

41. Wang, C.-Y. *et al.* 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*, 390–391 (2020).

42. Huang, Z. *et al.* Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection. *Inf. Sci.* **522**, 241–258, DOI: https://doi.org/10.1016/j.ins.2020.02.067 (2020).

43. Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8759–8768, DOI: 10.1109/CVPR.2018.00913 (2018).

44. Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2999–3007, DOI: 10.1109/ICCV.2017.324 (2017).

45. Zheng, Z. *et al.* Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 12993–13000, DOI: https://doi.org/10.1609/aaai.v34i07.6999 (2020).

46. Wojke, N., Bewley, A. & Paulus, D. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, 3645–3649, DOI: 10.1109/ICIP.2017.8296962 (2017).

47. Bewley, A., Ge, Z., Ott, L., Ramos, F. & Upcroft, B. Simple online and realtime tracking. *2016 IEEE Int. Conf. on Image Process. (ICIP)* DOI: 10.1109/icip.2016.7533003 (2016).

48. Guerrero-Gomez-Olmedo, R., Lopez-Sastre, R. J., Maldonado-Bascon, S. & Fernandez-Caballero, A. Vehicle tracking by simultaneous detection and viewpoint estimation. In *IWINAC 2013, Part II, LNCS 7931*, 306–316, DOI: 10.1007/978-3-642-38622-0_32 (2013).

49. Wen, L. *et al.* UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Comput. Vis. Image Underst.* DOI: 10.1016/j.cviu.2020.102907 (2020).

50. Niu, H., Lu, Q. & Wang, C. Color correction based on histogram matching and polynomial regression for image stitching. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, 257–261, DOI: 10.1109/ICIVC.2018.8492895 (2018).

51. Yu, G. & Morel, J.-M. Asift: An algorithm for fully affine invariant comparison. *Image Process. On Line* **1**, 11–38, DOI: 10.5201/ipol.2011.my-asift (2011).

52. Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1150–1157 vol.2, DOI: 10.1109/ICCV.1999.790410 (1999).

53. Fischler, M. A. & Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**, 381–395, DOI: 10.1145/358669.358692 (1981).

54. Wu, M., Zhang, C., Liu, J., Zhou, L. & Li, X. Towards accurate high resolution satellite image semantic segmentation. *IEEE Access* **7**, 55609–55619, DOI: 10.1109/access.2019.2913442 (2019).

55. Adams, R. & Bischof, L. Seeded region growing. *IEEE Transactions on pattern analysis machine intelligence* **16**, 641–647, DOI: 10.1109/34.295913 (1994).

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

57. Jung, H. *et al.* Resnet-based vehicle classification and localization in traffic surveillance systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, DOI: 10.1109/CVPRW.2017.129 (2017).

58. Wang, T., He, X., Su, S. & Guan, Y. Efficient scene layout aware object detection for traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, DOI: 10.1109/CVPRW.2017.128 (2017).

59. Hedeya, M. A., Eid, A. H. & Abdel-Kader, R. F. A super-learner ensemble of deep networks for vehicle-type classification. *IEEE Access* **8**, 98266–98280, DOI: 10.1109/ACCESS.2020.2997286 (2020).

60. Rezaei, M. & Azarmi, M. Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic. *Appl. Sci.* **10**, 7514, DOI: https://doi.org/10.3390/app10217514 (2020).

Appendix 1: Camera Calibration and Inverse Perspective Mapping

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

$$[x \;\; y \;\; 1]^T = \mathbf{K}[\mathbf{R}|\mathbf{T}]\,[X_w \;\; Y_w \;\; Z_w \;\; 1]^T \tag{31}$$

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:

$$\mathbf{K} = \begin{bmatrix} f*k_x & s & \mathrm{c}_x & 0 \\ 0 & f*k_y & \mathrm{c}_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{32}$$

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 $(\mathrm{c}_x, \mathrm{c}_y)$ are the principal points shifting the optical axis of the image plane.
$\mathbf{R}$ is the rotation matrix:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_c & -\sin\theta_c & 0 \\ 0 & \sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

where θ_c is the camera angle.
$\mathbf{T}$ is the translation matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{h_c}{\sin\theta_c} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

where h_c is the height of the camera.

These three matrices together $\mathbf{K}[\mathbf{R}|\mathbf{T}]$ are known as projection matrix $\mathbf{G} \in \mathbb{R}^{3\times4}$, so the transformation equation can be summarised as $[x \;\; y \;\; 1]^T = \mathbf{G}\;[X_w \;\; Y_w \;\; Z_w \;\; 1]^T$.

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:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix} \tag{35}$$

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

$$\Lambda((x,y),\mathbf{G}) = \left(\frac{g_{11}\times x + g_{12}\times y + g_{13}}{g_{31}\times x + g_{32}\times y + g_{33}}, \frac{g_{21}\times x + g_{22}\times y + g_{23}}{g_{31}\times x + g_{32}\times y + g_{33}}\right) \tag{36}$$