Determining the Speed of Vehicles from Video Images using Machine Learning

Francis Gurr
Supervised by Stefano Giani

***Abstract*—This report focuses on the use of road-side video cameras as a non-intrusive alternative to current intelligent transportation systems. The use of video to determine the speed of vehicles traveling on major roads was investigated. The report covers the tasks of object detection, tracking and camera calibration. A neural network, YOLOv3, was used for the purpose of object detection with an accuracy of 98% mAP. A simple Kalman filter was used to track the vehicles across the video frame, which worked well but struggled with longer periods of occlusion. The camera was calibrated using road markings in place of a reference object. The calibration method proved to be accurate, however, a constant error was introduced if the road markings were not consistent with the government specifications. The average vehicle speeds calculated were within the expected range and the model was able to run at real-time speeds.**

I. Introduction

IT has long been recognised that access to data can help improve a product or service. This extends to the transport industry. The Transport Data Initiative (TDI) was founded in 2016 and believes that 'improving the way we collect, store, and use data will help us deliver improved transport services while reducing the costs of delivery' [1]. Transport data can inform urban planning decisions and strategies and lead the way for smart cities and motorways. Additionally, road transport accounts for 22% of total UK emissions of carbon dioxide, and noise from road traffic affects 30% of people in the UK [2]. Intelligent transportation systems (ITS) can be used to help create and track initiatives that aim to solve such issues.

The most common form of ITS for vehicle detection uses inductive loops to detect axles in order to classify vehicles. These solutions are popular as they are highly accurate, however, a major disadvantage is that they are intrusive; they require the road to be closed during installation and maintenance [3]. This is costly and endangers workers lives. Therefore, there is a push to develop cheaper, non-intrusive solutions that can be deployed out-of-road.

Road-side video cameras could provide such an alternative. These are already used for the purposes of surveillance and, when combined with other sensing technologies, to enforce speed restrictions. Compared to induction loops, cameras would perform with less accuracy, as differing weather and lighting conditions add to the complexity of the system. However, they benefit from vastly lower installation and maintenance costs. This project was supported by Q-Free, who are developing a vehicle classification system using video images. This project investigated the use of video cameras to determine the speed of detected vehicles. This could add another unique product to the ITS market, providing clients with the flexibility to choose the data they require.

A. Object detection

The first step is to detect the vehicles in the video. Today, almost all state-of-the-art object detectors make use of neural networks. Before these methods were computationally feasible, background subtraction algorithms were commonly used. All background subtraction algorithms detect foreground objects in an image by comparing the image to a background model.

Neural networks are a set of algorithms that are loosely analogous to the structure of the human brain. The networks consist of layers of artificial neurons called perceptrons. These nodes combine a set of inputs with corresponding weights and apply an activation function in order to produce an output. The activation function controls whether the neuron is activated or dead. That is, whether or not the signal can pass through the node. Deep learning algorithms are then used to tune the weights to control which neurons should be activated so that the desired outputs are produced.

A convolutional neural network (CNN) is a type of neural network that is inspired by the visual cortex and is often applied to image tasks. A CNN consists of several convolutional layers that are used to extract features from an image. These features are then passed to a classifier.

The main drawback of CNNs is that they are computationally expensive. Therefore, using them with the sliding window approach for object detection is impractical due to the enormous number of window regions.

Faster R-CNN [4] provides a solution using a Region Proposal Network (RPN). The RPN generates a limited number of regions of interest to be evaluated by the CNN. This reduces the computational cost, significantly improving the performance.

You Only Look Once (YOLO) [5] offers further improvements in speed by removing the process of finding separate region proposals. Instead, it uses a single CNN on the entire input image only once, compared to around 2000 times for each region proposal in Faster R-CNN. Initially, YOLO suffered from a loss in accuracy, however, multiple improvements have led to YOLOv3 [6] which has a competetive accuracy without significant loss of speed. The industrial application of this project requires real-time performance, therefore, YOLOv3 was used for the task of object detection for this project.

B. Tracking

To determine the speed of a detected object its position must be tracked so that its location between frames can be compared.

Optical flow [7] is defined as the apparent motion of individual pixels on the image plane and can be used to track objects. The mathematics of optical flow can become rather difficult, and as individual pixels are being examined, the method is inefficient.

A common method used for tracking is the Kalman filter [8] as it is simple and effective. It is a recursive algorithm that is used to estimate the state of a process using a form of feedback control. There are also numerous other methods that build upon the Kalman filter, improving its performance.

The SORT algorithm proposed by Alex Bewley *et al.* [9] combines the Kalman Filter with the Hungarian algorithm [10] to produce a method that focuses on reliability. Both the basic Kalman filter and the SORT algorithm lack appearance information, therefore making them susceptible to identity switches, especially after longer periods of occlusion. The Deep SORT algorithm [11] tackles this problem by integrating appearance information from the output of the CNN, with only a minimal reduction in speed.

For this project, a basic Kalman filter was used due to its simplicity.

C. Camera Calibration

To be able to calculate the speed in real world units the camera must first be calibrated. Complete camera calibration consists of distortion correction and camera resectioning; mapping the 2D image plane to the 3D world scene. In this project only camera resectioning was considered and any distortion effects were assumed to be negligible.

There are many camera calibration methods due to the number of possible combinations of known and unknown camera parameters. However, despite this variation, many popular solutions can be categorised into two main approaches. In the first approach, some parameters are found from the camera specifications and by accurately measuring the camera set-up to find its tilt, pan, swing and translation. This has its drawbacks as performing accurate measurements can often be difficult or impractical, especially if the camera is in an inaccessible location. The second approach aims to derive the parameters using a reference object of known size or known geometric features in the camera scene.

One commonly used method involves the use of a chessboard pattern for calibration [12]. The exact, regular pattern enables the camera to be reliably calibrated to remove both radial distortion and find the camera parameters. However, this method is only practical for small-scale scenes, as the pattern must fill a substantial area of the image. Therefore, to calibrate a road-side camera, the chessboard would have to be impractically large.

Other methods use a stereo camera system where two cameras are used to observe the same scene from different positions [13]. This method mimics stereopsis, the biological process used by most animals to ascertain depth. However, the method requires either accurately installing two cameras with a precise separation distance, or the use of a specialised stereo camera. This increases the complexity and cost of the solution.

Sochor *et al.* [14] propose a method of particular interest, it is fully automatic and requires no user input. Vehicles are detected and tracked and the motion of the traffic in each direction is averaged and assumed to be along a straight line. The camera parameters can then be calculated using the vanishing points that are found at the intersection of these lines. The scene scale is inferred by comparing known 3D models of frequently passing cars with the detected vehicle on the image plane. This method is complex and out of the scope of this project, but shows great potential, achieving a relative error of just 1.39%.

Fung *et al.* [15] and He and Yung [16] propose different methods for calibrating the camera using standard road markings as reference features. These methods were used to calibrate the camera for this project as they are simple, effective and require no knowledge of the camera or its set-up. Additionally, standard road marking specifications are widely available and no other reference objects were required.

II. Theory

A. YOLOv3

YOLOv3 is a fully convolutional neural network (CNN) that is extremely fast since it does not use region proposals, instead it processes the entire image only once.

YOLOv3 uses a feature extractor called Darknet-53 [6], shown on the top row of Figure 1, that consists of 53 convolutional layers, each followed by a batch normalization layer and an activation function. Most of the 3×3 convolutional layers are preceded by a 1×1 convolution which is used to perform reduction in the filter dimension so that the 3×3 convolutions are less computationally expensive.

The batch normalisation layer standardises the inputs of the activation function to have a mean close to zero and a standard deviation close to one. This has multiple benefits, including speeding up the training as the network should converge more quickly. It also serves to simplify the network as other regularisation methods such as dropout can be removed.

The activation function used is the leaky variant of the Rectified Linear Unit (ReLU). ReLU is defined as $y(x) = \max\{0, x\}$ and is commonly used as it is computationally simple, converges quickly due to its linearity and has distinct 'dead' and 'active' or 'firing' regions. The leaky variant has a small slope of $y = 0.01x$ for all negative values instead of being completely 'dead' at zero. This allows for a neuron to recover from being 'dead' and can allow for faster learning rates without ending up with large numbers of 'dead' neurons.

The feature extractor also utilises residual blocks. These provide shortcuts between convolutional layers, adding the activation outputs. In large networks the gradient can often degrade to zero as the accuracy saturates, making gradient descent extremely slow. The shortcut connections between the layers remedy this problem.

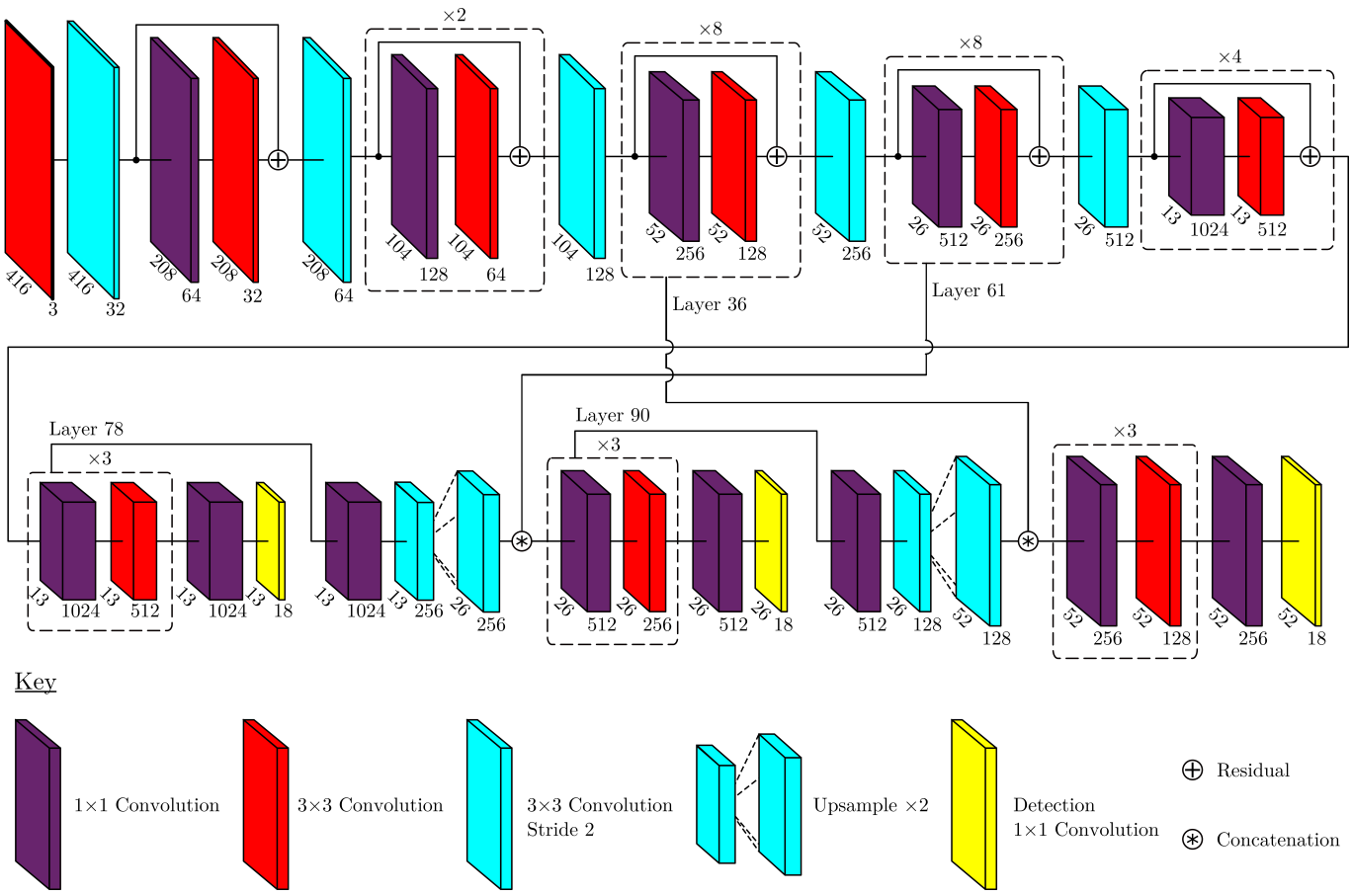


Fig. 1: YOLOv3 architecture, with Darknet-53 on the top row

The feature maps are downsampled using a convolutional layer with stride two. This is better than using pooling as it prevents the loss of low-level features.

The features learned by the convolutional layers are then passed to a classifier which makes the detection predictions using 1×1 convolutions at three different scales as shown in Figure 1. Between the classifier at each scale, the convolutional layer is upsampled and concatenated with the feature map of a previous layer.

At each scale the image is divided into an $S\times S$ grid and the detection of an object is determined by the grid cell that contains the centre of the object. The YOLOv3 predictions are log-space transforms that are applied to a set of anchor boxes, or bounding box priors, to obtain a set of bounding box predictions per grid cell. Using anchor boxes allows the network to predict the bounding boxes much faster. In this case, three anchor boxes are used and therefore three bounding boxes are predicted per grid cell. For each anchor box, C class scores are predicted as well as five attributes that describe the objectness score and the box centre and size. Objectness is the likelihood of there being an object in the box. Class score is the probability that the object belongs to a certain class. The final bounding box is computed from the network outputs t_x, t_y, t_w and t_h as follows:

$$b_x = \sigma(t_x) + c_x \tag{1}$$
$$b_y = \sigma(t_y) + c_y \tag{2}$$
$$b_w = p_w e^{t_w} \tag{3}$$
$$b_h = p_h e^{t_h}. \tag{4}$$

The coordinates of the centre of the final bounding box are (b_x, b_y) and its width and height are b_w, b_h respectively. (c_x, c_y) are the coordinates of the top left corner of the grid cell and p_w, p_h are the width and height of the anchor box respectively. The sigmoid function σ restricts the output of the centre coordinates to be between zero and one to ensure it lies within the grid cell.

For an input image of size 416×416px, YOLOv3 predicts a total of 10647 bounding boxes, therefore the output must be processed first in order to get a reasonable set of results. Initially, the boxes are filtered based on their objectness and confidence scores and boxes below a threshold are ignored. Non-maximal suppression is then used to remove multiple detections of the same object. This is implemented by selecting the box with the highest score, then calculating its intersection over union (IOU) and removing any overlapping boxes that fall below the IOU threshold.

B. Kalman Filter

The Kalman filter [8] is a recursive algorithm used to estimate the state of a process using a form of feedback control. It is very useful for tracking applications where the aim is to predict where an object will next be found. The filter makes the assumptions that the model of the process is linear and that the noise is white Gaussian.

The filter can be broken down into two steps; predict and update. During the predict step the optimal prediction of the current state and its expected variance are calculated. After a measurement, the update step combines the values and variances of the measurement and predicted state with a gain. If the measurement is noisy and has high variance, the gain is small and therefore reduces the weight of the measurement value. If the predicted variance is large, the gain will be large and so the measurement will be weighted heavily.

For this project, the Kalman filter uses the bounding boxes predicted by the network as the measurements in the update step.

C. Camera Calibration

The process of camera calibration is necessary in order to determine the vehicle speed in real-world units. It is used to find the parameters of the pinhole camera model that most closely approximate the camera the image was taken with. These parameters are represented by the camera projection matrix

$$P = K \begin{bmatrix} R & T \end{bmatrix} \tag{5}$$

where K consists of the intrinsic parameters and $\begin{bmatrix} R & T \end{bmatrix}$ corresponds to the extrinsic parameters, where R is the rotation matrix and T is the translation matrix [17].

The intrinsic parameters are described by the matrix

$$K = \begin{bmatrix} rf & s & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where r is the aspect ratio, f is the focal length in pixels, s is the skew factor and (u_0, v_0) are the coordinates of the principal point [18]. The skew factor describes the amount of shear distortion in the projected image, and the principal point is the intersection between the image plane and the line that passes through the pinhole of the camera. Often, the principal point is assumed to be $(0, 0)$, the skew factor is assumed to be 0 and the aspect ratio is set to 1, reducing the intrinsic parameters to the focal length.

The extrinsic parameters represent the position of the camera in the 3D world scene [19]. This is described by the translation and rotation of the camera from the origin. Given the general camera model in Figure 2, the horizontal rotation is given by the pan angle p, the vertical rotation by the tilt angle t, and the rotation along the camera's optical axis is given by the swing angle s. The location of the camera is given by the world coordinates $(X_{CAM}, Y_{CAM}, Z_{CAM})$.

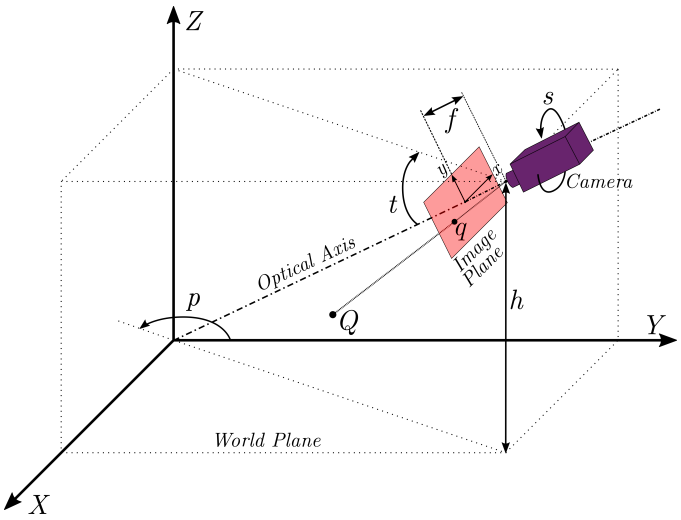


Fig. 2: General camera model (Based on [19])

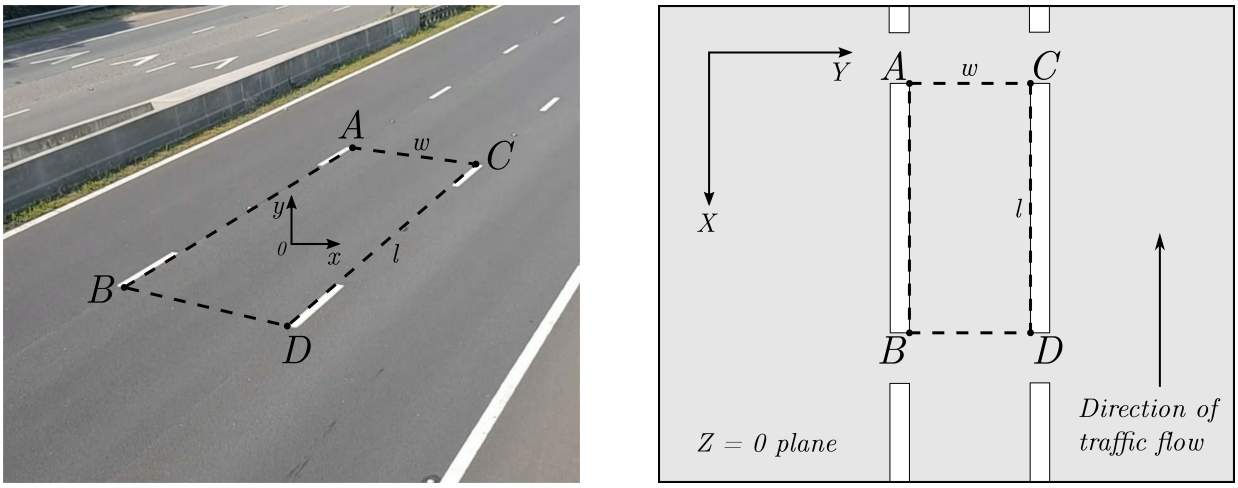
Knowledge of the camera matrix allows for coordinates in the image plane to be projected onto the 3D world scene as shown.

$$\lambda \begin{bmatrix} x, & y, & 1 \end{bmatrix} = \begin{bmatrix} X, & Y, & Z, & 1 \end{bmatrix} P \tag{7}$$

where λ is the scale factor, (x, y) are the coordinates of a point on the image plane and (X, Y, Z) are coordinates of the corresponding point in the world scene.

The proposed calibration method is based on the methods presented by Fung and Yung [15] and He and Yung [16], and uses a rectangular pattern from road markings to calculate the camera parameters. The 3D world plane is defined with the X-axis along the direction of the road, the Y-axis perpendicular to the road and the Z-axis perpendicular to the surface of the road. It is assumed that the road is straight and flat and that all points on the road are on the $Z = 0$ plane in the 3D world scene. Four points on the road surface are selected using the road markings as a reference to form a rectangle $ABCD$ with the sides $\overrightarrow{AB}$ and $\overrightarrow{CD}$ parallel to the X-axis and the sides $\overrightarrow{AC}$ and $\overrightarrow{BD}$ parallel to the Y-axis in the world scene. The width and length of the rectangle are w and l respectively. This rectangle is shown in Figure 3 as seen in both the image plane and the world scene.

The method has the advantage that it can be used to calculate all the necessary camera parameters without requiring any prior knowledge of the camera setup. The method presented by Fung and Yung (FY) calculates the parameters using the width of the rectangle and the two vanishing points which are found from the intersection of the parallel lines. However, FY suffers from ill-conditioning at pan angles close to 90 degrees. This is because as the pan angle tends to $90n$ degrees, where $n \in \mathbb{Z}$, the second vanishing point tends to infinity as one set of lines becomes parallel in the image scene. He and Yung (HY) propose an alternative method to calculate the parameters in these cases, using just one vanishing point and the width and length of the rectangle. This method cannot be used for every case as it suffers from large error in certain cases and therefore a combination of the methods must be used.



(a) Viewed in the image scene

(b) Viewed top down in the 3D world scene

Fig. 3: Example of road markings used for calibration (Based on [15])

Table I presents the three different cases and the calibration method that is most suitable. The method used in Case 3 is the same as the standard HY method but with the labels of the rectangle vertices temporarily rotated by 90 degrees, effectively transforming the camera by a pan angle of 90 degrees in order to calculate the other parameters.

TABLE I: Case selection

Case	Pan Angle p (degrees)	Calibration Method
1	$90n + 30 < p < 90n + 60$	Fung & Yung (FY)
2	$180n + 60 < p < 180n + 120$	He & Yung (HY)
3	$180n - 30 < p < 180n + 30$	HY Rotated (HY′)

where $n \in \mathbb{Z}$

III. Methods

A. Preprocessing

The dataset was provided by Q-Free and consisted of approximately $30,000$ images of UK A-roads and motorways, of which around $15,000$ were labeled for nine different classes of vehicles as well as cyclists and pedestrians. The dataset also included images from an IR camera at night, some images during adverse weather conditions and a few poor quality images. These images were manually removed to simplify the dataset. A Python script was then used to discard the unlabeled images, remove the pedestrian and cyclist classes and replace the other classes with one singular 'vehicle' class. The Python script also split the dataset into training, test and validation sets at a ratio of 7:2:1.

Another Python script was used to create videos from the still images in the dataset. These videos were used to investigate the rate at which the model could perform the detections. Q-Free also supplied a video file, however, as this was handheld footage it was considerably shaky. The video was stabilised as best as possible using video editing software, as the camera calibration is only valid for a fixed viewpoint . Despite this, only one ten second section of the video was deemed usable.

B. Training

The darknet framework from Alexey [20] was used to configure and train the network. The size of the network chosen was 416×416, which was the maximum size capable of running on the machine used. The framework included data augmentation tools, which were used to randomly vary the saturation, exposure, hue, resolution and aspect ratio of the images, therefore artificially increasing the size of the dataset. The initial convolutional weights used were pre-trained on Imagenet [21]. The three anchor boxes were calculated using k-means clustering on the dataset. The labeled boxes were grouped into three clusters, and the mean of each cluster formed the shape of the anchor boxes. The burn-in rate was set to 1000 and the learning rate was started at 0.001 and decreased by 0.1 at 4800 iterations and again at 5400 iterations. This was to stop the model from rapidly converging to an incorrect solution. The hardware used for training was an Nvidia GTX 1080 with 8GB of VRAM.

C. Calibration tools

A calibration tool was developed using Python and OpenCV to enable the user to easily calibrate a camera [22]. It is assumed that the camera is in a fixed position and that distortion effects, such as barrel distortion, were negligible. It is also assumed that the road is flat and straight, and that the road markings used for calibration have parallel features and conform to the government road specifications. The user first selects the vertices of a rectangle $ABCD$ by clicking on an image from the camera. The rectangle should use road markings for reference as shown in Figure 3. To reduce the user error, the points can be inputted multiple

times to form an average location for each. The width of the rectangle is specified and then the software calculates the camera parameters using Fung and Yungs method (FY). The dimensions of the road markings on UK roads can be found in the Traffic Signs Manual from the Department for Transport [23]. The pan angle is then used to check whether the parameters should be recalculated using He and Yungs (HY) method or HY′ with the $ABCD$ labels rotated by 90 degrees as shown in Table I. The camera parameters are then written to file to be used by the main program to calculate the speeds.

D. Calculating speed

The tracking and speed calculations are implemented using Alexey's C++ API [20] and a modified version of his C++ example file [22]. The tracking is performed using the OpenCV Kalman filter. This standard implementation of the Kalman filter did result in some erroneous behavior as the originally detected bounding boxes would only be used to update the filter during the update step and were then discarded. Therefore, the location of the bounding box as predicted by the network was lost and replaced by that predicted by the Kalman filter which was less accurate, hence the speed measurements were affected. To remedy this behavior, rather than changing the API, the Kalman filter predictions were matched to the bounding box predictions of the network. For each network predicted box, it was compared to every Kalman prediction and a score for each pairing was calculated. This score is given by the intersection area over the area of the network predicted box. For the pair with the highest score the tracking ID from the Kalman filter was applied to the corresponding network predicted bounding box.

Next, the average speed of each vehicle passing the view of the camera was calculated. It is assumed that the vehicles are moving at a constant speed. First, the camera parameters are loaded from the file generated by the calibration tool. For each detected vehicle, the corner of the bounding box closest to the ground plane is selected. The corner coordinates (x_q, y_q) in the image plane are then converted to a set of corresponding coordinates (X_Q, Y_Q) on the $Z = 0$ plane in the real world scene using the following transformations

$$X_Q = \frac{\mu_s h \cos p + \left[\frac{\nu_s h \sin p}{\sin t}\right]}{x_q \cos t \sin s + y_q \cos t \cos s + f \sin t} \tag{8}$$

$$Y_Q = \frac{\mu_s h \sin p - \left[\frac{\nu_s h \cos p}{\sin t}\right]}{x_q \cos t \sin s + y_q \cos t \cos s + f \sin t}, \tag{9}$$

where

$$\mu_s = x_q \cos s - y_q \sin s \tag{10}$$

$$\nu_s = x_q \sin s \ + \ y_q \cos s. \tag{11}$$

The first time a new vehicle is detected, its tracking ID, initial coordinates (X_Q, Y_Q) and the current frame number f_q are recorded. Each time the same vehicle is detected, its distance d from (X_Q, Y_Q) and the elapsed number of frames t_f since f_q are calculated. Using these, the average speed v of the vehicle from when it was first detected is

$$v = \frac{d}{t_f \cdot \frac{1}{FPS}} = d \cdot \frac{FPS}{t_f} \tag{12}$$

where FPS is the frames per second of the video feed.

IV. Complications

Due to the Covid-19 outbreak, the final set of results could not be obtained. All of the necessary prior work was completed, however the following results could not be collected due to the shutdown:

- the accuracy of the model on a separate validation set;
- an average vehicle speed per lane for a longer fully stabilised video;
- the average speed of the model in frames per second, especially whilst not simultaneously displaying a video output stream, as would be expected under normal operation;
- the performance of the model across different network sizes;
- the performance of the network across a variety of camera angles.

Therefore, in the following results section only the observations made on a handful of test-time outputs are discussed.

V. Results

A. Object detection

The model achieved very high accuracy for the task of object detection with 98% mAP. However, as the datasets are not the most representative of the true environment, the model will need to be retrained with different data. As the dataset used was simplified to only contain images from clear sunny days, the model would suffer during adverse weather conditions. The model would also not function overnight using IR camera images, it would need to include these in the dataset or switch to a separate model overnight. Additionally, the dataset lacked in variety. There were not many different camera angles. Additionally, the images did not contain many different vehicle classes, as shown in Figure 4. This resulted in poor accuracy for rarer vehicle types, such as motorcycles. The dataset also did not contain labels for all of the vehicles in each image, in particular distant vehicles were neglected. This led to the model only detecting and rating its performance for objects in the foreground. For the purpose of measuring the speed of the vehicles it would be especially useful to detect them as soon as possible. This would increase the time and distance traveled by the detected vehicle across the frame, therefore improving the speed averages.

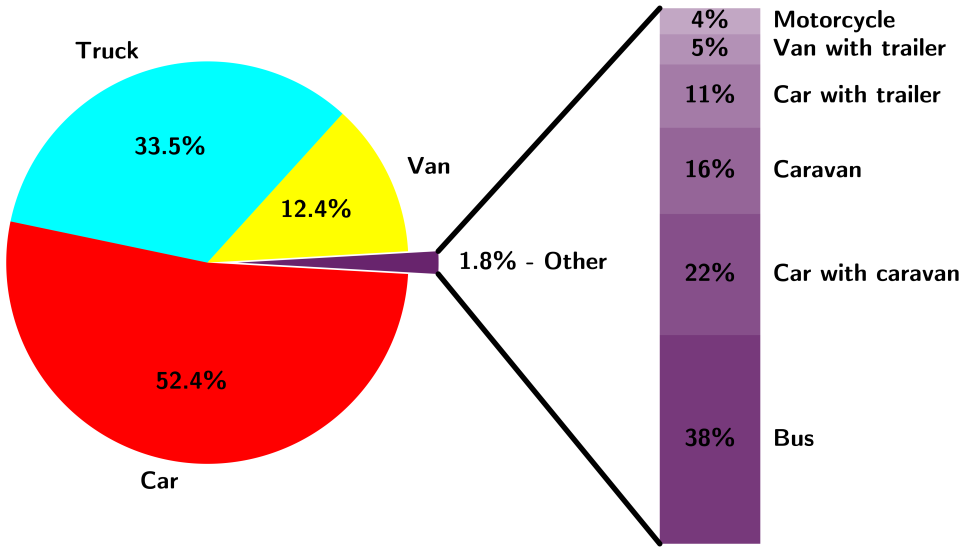


Fig. 4: Vehicle class distribution in the dataset

For a network size of 416×416, using a HD 1920×1080px video as input, the model reached a peak speed of 42 FPS while displaying the output detections as a video stream. The majority of cameras record at 30 FPS, therefore the model is more than fast enough for real-time detection, especially considering that in practice the output would not be in video format and would therefore be even faster. However, the accuracy of the detections for the video footage was lower than that of the image test set, as the camera angle was different to those in the training dataset.

B. Tracking

The Kalman filter successfully tracked vehicles across the video frame in most cases. However, periods of occlusion caused issues as the filter would not recognise the object again. This issue was most prevalent during periods of dense traffic. Measures can be taken to reduce the effects of occlusion by carefully selecting the camera position. The higher the camera is installed, the less occlusion will occur. The camera angle relative to the direction of the road will also have an effect. The closer the camera is to being perpendicular to the road, the more vehicles will obstruct those in other lanes. The closer the camera is to being parallel to the road, the more likely that vehicles are obstructing vehicles in the same lane. Having the camera close to parallel would also mean that the vehicles would rapidly reduce in size, because the vanishing point would be closer. Therefore the size of the region where cars are successfully detected is reduced. For these camera angles, the Kalman filter caused some vehicle identities to jump across the road nearer the vanishing point. However, this was easily remedied by cropping out the other side of the road.

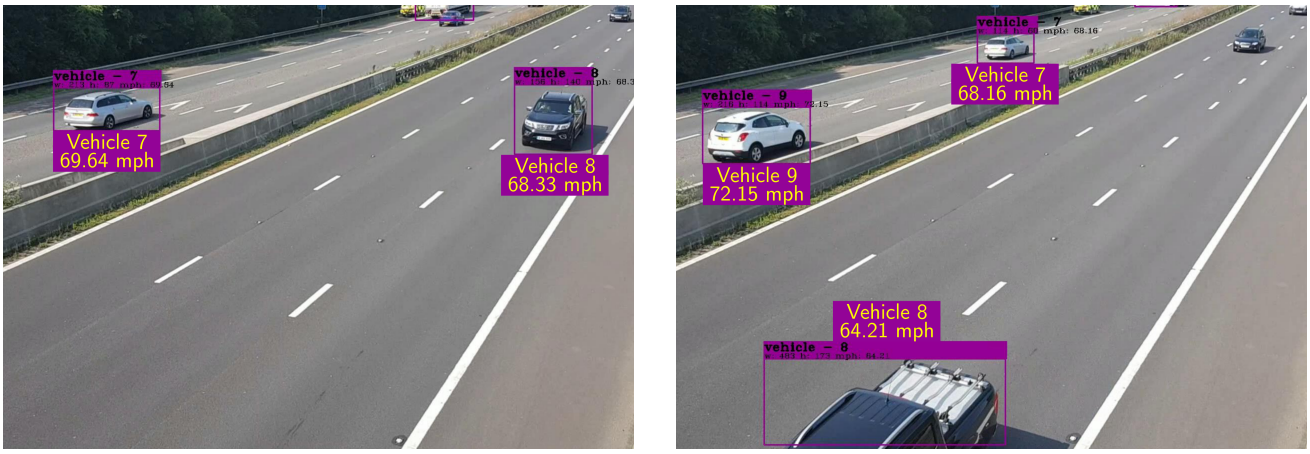
C. Vehicle speed

The calculated speeds were in the region of expected speeds for the road and vehicle class, however, there were no ground truth measurements for comparison. The speed measurements for vehicles in the overtaking lanes were faster than for the vehicles in the slow lane, as expected. This can be seen in the two frames from the video output in Figure 5. Vehicle 9 is in the fast lane and traveling faster than vehicles 7 and 8. All three vehicles are also traveling close to 70mph which is expected for this type of road.

The camera calibration method was accurate, however, it relied on knowing the accurate dimensions of the road markings. Occasionally the road markings deviated from the government road specifications and in these cases the speed measurements were all uniformly affected due to the constant calibration error. The camera calibration was also only valid for short periods of time, as the video was not taken with a tripod and therefore the camera was not in a truly fixed position.

Using the corner of the bounding box closest to the road surface did not result in the most accurate solution. This is because YOLOv3 best predicts the bounding box center and only fits the box shape using an anchor box, resulting in loosely fitting bounding boxes. Therefore, the corner of the box was never perfectly on the $Z = 0$ plane in the world scene. The corner of the bounding box would also remain relatively stationary while the object was coming into view at the edge of the frame. This would result in errors in the average speed.

For a network size of 416×416, using a HD 1920×1080px video as input, the model reached a peak speed of 36 FPS whilst both displaying the output detection as a video stream and performing the tracking and speed calculations. Using the road markings, the length of the road visible in Figure 5 can be estimated to be 60m. Therefore, a vehicle traveling at 70mph will cross the video frame in just under two seconds and will be captured by the camera in approximately 60 frames.



(a) Frame 50

(b) Frame 77, approximately one second later

Fig. 5: Two successive captures from the video output

As the model's speed is faster than the video's 30 FPS, all occurrences of the vehicle will be used. For calculating the vehicle speed, an absolute minimum of two occurrences, or just over 1 FPS, is necessary. However, this would not be sufficient to track the vehicle, which is necessary to calculate the speed. Therefore, the more occurrences the better and currently the model is using the the maximum number of captures as it is running faster than the camera.

For real-time application, the network would be running on less powerful hardware. However, the model would not be required to display the output as a video stream, which would further increase the computational performance. Therefore, the network shows very strong potential to perform at the speeds required for real-time application.

VI. Conclusion and Further Recommendations

This work shows that using video images has the potential to be a successful non-intrusive solution for calculating the speed of traffic flow. The accuracy rates achieved for the detection are high and most importantly the system has been shown to be capable of running at real-time speeds. To further improve the system, the model should be trained on a more varied dataset. It should include images from a larger variety of camera angles, various weather conditions and include more images of less common vehicles, such as motorcycles. Additionally, there should be images, or a separate model, for overnight IR camera images. Most importantly, all visible vehicles in an image should be labeled to allow the model to be trained for every vehicle in view.

To improve the tracking, the Deep SORT algorithm [11] should be used. By including the object features, occluded objects would be tracked much more successfully.

The camera calibration method is very good in principle. However, its accuracy relies on the road markings and this is not reliable in practice as they do not always conform to the government road specifications. The calibration method presented by Sochor *et al.* [14] shows promise, as it is fully automatic, requires no user input and does not rely on any reference markers.

To improve the accuracy of the speed measurements, 3D bounding boxes should be used. The center of the ground plane of the 3D bounding box would be a much more accurate marker than the corner of the 2D bounding box that is closest to the $Z = 0$ plane. Using 3D boxes would also open up the possibility of being able to output the size of the object without much additional effort. The accuracy of the model should also be investigated using ground truth speed data.

Finally, the performance of the system should be tested using hardware that is more representative of the final application to investigate how it performs when implemented.

Acknowledgement

The author would like to thank Q-Free for supporting the project by supplying the data and providing help along the way. Thanks is also given to Stefano Giani, whose guidance and supervision was invaluable.

References

[1] The tdi website: About us. [accessed: 22/10/2019]. [Online]. Available: http://transportdatainitiative.com/about-us/

[2] Environmental protection uk: Impacts of car pollution. [accessed: 22/10/2019]. [Online]. Available: https://www.environmental-protection.org.uk/policy-areas/air-quality/air-pollution-and-transport/car-pollution/

[3] R. Avery, Y. Wang, and G. Rutherford, "Length-based vehicle classification using images from uncalibrated video cameras," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC'04)*, Washington, WA, USA, Oct. 2004, pp. 737–742.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, Jun. 2017.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, USA, Dec. 2016, pp. 779–788.

[6] A. F. J. Redmon, "Yolov3: An incremental improvement," University of Washington, Tech. Rep., apr 2018.

[7] B. Horn, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, Aug. 1981.
[8] R. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, vol. 82, pp. 35–45, Mar. 1960.
[9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE International Conference on Image Processing (ICIP'16)*, Phoenix, AZ, USA, Aug. 2016, pp. 3464–3468.
[10] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, Mar. 1955.
[11] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE International Conference on Image Processing (ICIP'17)*, Beijing, China, Sep. 2017, pp. 3645–3649.
[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, p. 1330–1334, Nov. 2000.
[13] G. Xu, L. Chen, and F. Gao, "Study on binocular stereo camera calibration method," in *Proc. IEEE International Conference on Image Analysis and Signal Processing (IASP'11)*, Hubei, China, Oct. 2011.
[14] J. Sochor, R. Juranek, and A. Herout, "Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement," *Computer Vision and Image Understanding*, vol. 161, pp. 87–98, Jun. 2017.
[15] G. Fung, N. Yung, and G. K. H. Pang, "Camera calibration from road lane markings," *Optical Engineering*, vol. 42, pp. 2967–2977, Oct. 2003.
[16] X. C. He and N. Yung, "New method for overcoming ill-conditioning in vanishing-point-based camera calibration," *Optical Engineering*, vol. 46, Mar. 2007.
[17] Z. Zhang, T. Tan, K. Huang, and Y. Wang, "Practical camera calibration from moving objects for traffic scene surveillance," *IEEE Transaction on Circuits and System for Video Technology*, vol. 23, pp. 518 – 533, Mar. 2013.
[18] Y. Zheng and S. Peng, "A practical roadside camera calibration method based on least squares optimization," *IEEE Transaction on Circuits and System for Video Technology*, vol. 15, pp. 831 – 843, Apr. 2014.
[19] G. Fung, N. Yung, and G. Pang, "Camera calibration from road lane markings," *Optical Engineering*, vol. 42, pp. 2967–2977, Oct. 2003.
[20] AlexeyAB, "Darknet," GitHub, [accessed: 15/3/2020]. [Online]. Available: http://transportdatainitiative.com/about-us/
[21] P. J. Reddie, "Darknet pre-trained weights," [accessed: 15/3/2020]. [Online]. Available: https://pjreddie.com/media/files/darknet53.conv.74
[22] F. B. Gurr, "Final year project code," GitHub, [accessed: 25/4/2020]. [Online]. Available: https://github.com/Francis-Gurr/Vehicle-Speed-YOLO
[23] D. for Transport, "Chapter 5: Road markings," *Traffic Signs Manual*, 2019.