# Real Time Traffic Detection of The Wrong Side Vehicle

**Anish kumar**[1]
[1] MCA Student, Department of Computer Application, Quantum Universiry, Roorkee, India
**Shivank Tyagi**[2]
[2] MCA Student, Department of Computer Application, Quantum Universiry, Roorkee, India
**Jamshed Alam**[3]
[3] MCA Student, Department of Computer Application, Quantum Universiry, Roorkee, India
**Kapil Dev**[4]
[4] MCA Student, Department of Computer Application, Quantum Universiry, Roorkee, India
**Sumit Raghuwanshi**[5]
[5] MCA Student, Department of Computer Application, Quantum Universiry, Roorkee, India

**Featured Application: This study introduces a deep learning-based application that detects moving violations in vehicles and performs well in a variety of illumination scenarios. For closed-circuit television (CCTV) cameras on the road, the method is a useful aid.**

**Abstract:** This paper is about designing a real-time Intelligent Transport System (ITS) that detects wrong-way drivers on dangerous roads. Whenever such a car reaches the zone under surveillance, the system immediately detects it using just one CCTV camera. The detection device sends a warning signal to the drivers and alerts the monitoring centre simultaneously. The core of the proposed method consists of three key processes: detection, tracking, and validation, which are achieved through video imaging techniques. Detecting vehicles in video frames is achieved with a Deep Learning method named You Only Look Once (YOLOv3), trained on a custom dataset.
The system tracks the vehicles' movements over time by means of linear quadratic estimation, known as Kalman filtering. In the final step, the trajectory of vehicles is found by an "entry-exit" algorithm that has been shown to recognize drivers moving in the wrong direction with an accuracy of 91.98%.

**Keywords:** linear quadratic estimation (LQE), YOLOv3, intelligent transportation systems (ITS), and convolutional neural networks (CNNs)

## 1. Introduction

As the fourth industrial revolution approaches, intelligent transport systems (ITS) are becoming more and more important for improving driver efficiency and safety. In addition to maximising the use of surveillance technology, vision-based ITS seeks to gather useful and accurate traffic data. Korea has one of the greatest camera-to-population ratios in the world, with over eight million closed-circuit television (CCTV) cameras deployed by 2018, according to the National Information Society Agency. Even while CCTV footage cannot handle every traffic infraction, government officials are using it more and more to discover serious infractions that are backed up by documentation, witnesses, and proof. Driving the wrong way down a road usually occurs for a number of reasons, such as being distracted or confused, failing to see traffic signs or pavement markings, purposefully breaking the law, etc. Despite the inventive nature of contemporary pavement delineations and signage, car accidents still occur because they may not be sufficient to warn drivers of wrong-way entrance.

Although human driving behaviour is nearly impossible to control, it is crucial to identify and comprehend anomalous conduct in typical traffic situations in order to avoid major auto accidents. The identification task is being carried out by employees of monitoring businesses. However, the demand for automated intelligent software has increased due to the notable rise in camera devices. The three main kinds of wrong-direction detection methods are sensor-based, radar-based, and video imaging-based detection. The goal of this research is to detect (with a single video camera) vehicles entering traffic in the incorrect direction onto the highway system and to automate the recognition of moving violations, thereby reducing the need for human contact (Figure 1).

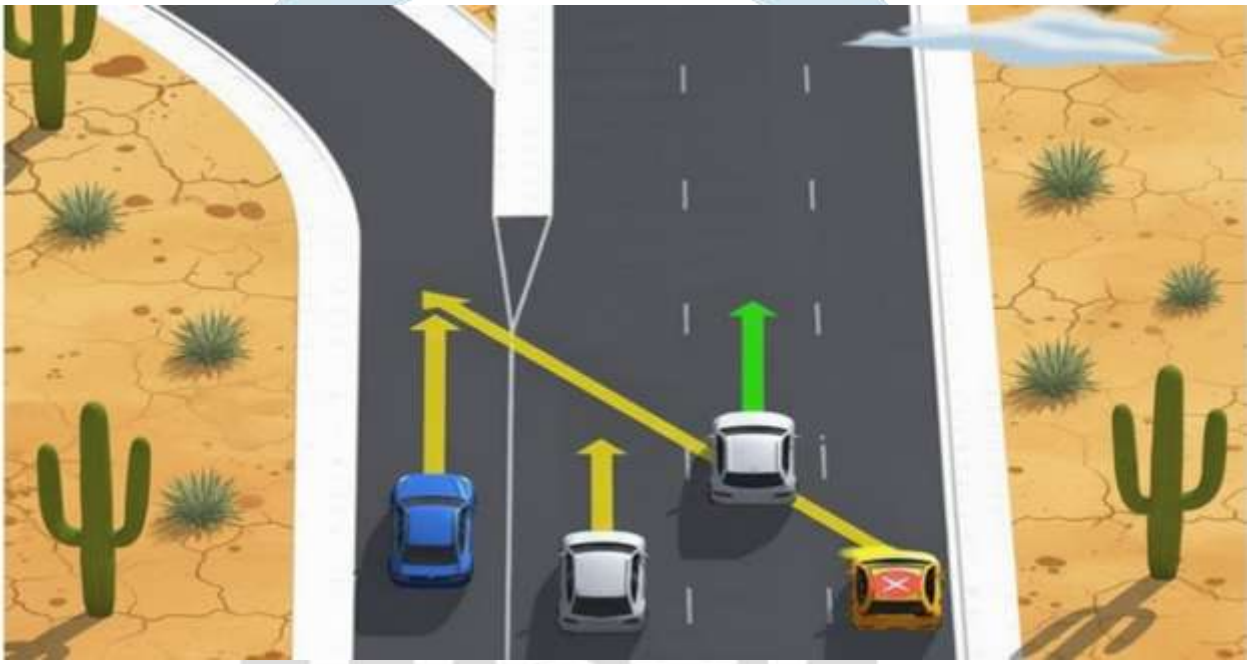

**Figure 1.** Situation involving wrong-direction detection (the triangle represents the area of view of the camera).

The three main computer vision problems of vehicle localisation, tracking, and direction determination are addressed by our suggested method, which makes use of video imaging-based detection. By saving time, minimising the need for large machinery, and getting rid of faulty items, vision-based learning is an economical method that lowers labour costs overall.

## 2. Background

The three main computer vision problems of vehicle localisation, tracking, and direction determination are addressed by our suggested method, which makes use of video imaging-based detection. By saving time, minimising the need for large machinery, and getting rid of faulty items, vision-based learning is an economical method that lowers labour costs overall.

### 2.1 *Vehicle Detection and Segmentation*

#### 2.1.1 Detection Using Background Subtraction

Background subtraction is one of the popular techniques to obtain a foreground mask in case of fixed cameras. It generates a foreground mask of moving objects extracted from the current frame and the background model which is considered as a static scene component. Every detected motion is considered foreground. Background initialization and background updating constitute two main procedures of BS. In this segment, OpenCV implementations of BS algorithms are being discussed. Some well-known background subtraction methods in

OpenCV are MOG2-Gaussian Mixture Model-based background/foreground segmentation method [2,3] and KNN algorithm [1]. The BS method has always been unstable, yielding false positives because it usually merges several close cars into one object (Figure 2). Furthermore, the method is extremely sensitive to illumination changes and often gives unacceptable errors during the night-time.

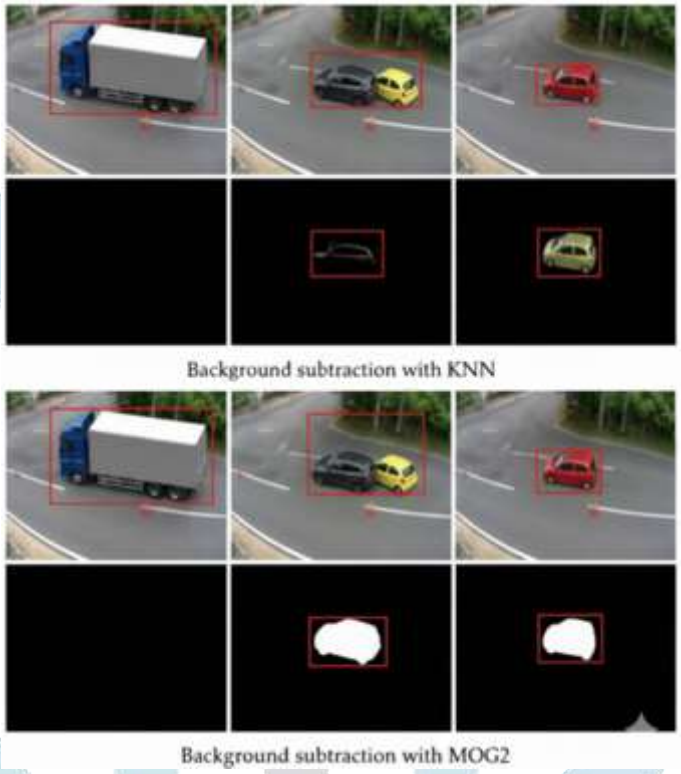


**Figure 2.** Comparison between two background subtraction methods. KNN: k-nearest neighbor; MOG2: Gaussian mixture-based background/foreground segmentation.

### 2.1.2 Optical Flow

An instance of a prevalent machine learning method for monitoring the movement of articles is characterized by optical flow, which discovers autos. Besides that, in this particular area, the Lucas-Kanade algorithm is the prevailing system [4]. The point being that a pixel's color or light intensity doesn't differ as it goes from a frame to another frame. The algorithm also assumes that items are only slightly and locally moved from one frame to the next. The reliability of motion information at the local level is also objectively measured by it. By assessing the textural properties of particular regions and improving the "good feature" selection criteria for dependability, Shi and Tomasi [5] improved tracking performance. The Kanade–Lucas–Tomasi (KLT) tracker [6] was later developed as a result of this advancement (see Figure 3). But according to Barron et al. [7], optical flow methods are extremely susceptible to noise and variations in object illumination. In situations where there are changes in brightness intensity or object occlusions, these techniques frequently lose their dependability. Considering our task's lowest error rate criteria, optical flow might not be the best option for detection. In such settings, its sensitivity to variations in weather and lighting could result in inaccurate readings.

Monteiro et al. [8] suggested a three-stage optical flow-based method for detecting drivers who are driving in the wrong direction. These processes include learning, detection, and validation. For the learning phase focused on vehicle detection, they merged Gaussian mixture models with the Lucas–Kanade optical flow technique [9]. Their technique detected cars travelling in the incorrect direction by using 320 × 240 pixel

images at 33 frames per second. Tests conducted in various environments showed the system's benefits, including its capacity to identify vehicles going in the wrong direction, its speedy processing speed, and its adaptability to changes in lighting and weather. However, the accuracy did not support the genuine benefit of their strategy.

**Figure 3.** Detection with optical flow using the Lucas–Kanade method.

### 2.1.3 Convolutional Neural Network (CNN)-Based Detection Methods

Convolutional Neural Networks (CNNs) are employing for the image analysis and understanding to the great extent. One of the main advantages of these networks concerns a small size of the learning parameters that drastically reduces the time it takes to learn and the volume of data that must be collected to build the CNN. A convolution operation is the process of moving a sliding window across an image and computing the weighted sum of the pixel values. By using a weighted matrix, this process produces a transformed version of the image. CNNs is a very powerful method used to locate the vehicle in an image with high precision. The advanced ones are based on the Single Shot MultiBox Detector (SSD), Faster Regional CNN (Faster R-CNN), and You Only Look Once (YOLO) patterns as a role model in this wave. However, the methods are largely the same in terms of average crankiness in detection but their computational loads are very different. Besides the fact that Faster R-CNN was at the top of the list by adding a region proposal network to the CNNs category, this method has been the best performing in the Pascal Visual Object Classes (VOC) and Common Objects in Context (COCO) challenges in 2012 and 2015. However, we ended up picking the YOLOv3 network over the others, due to its superior speed performance, even though the technology was developed in 2018 and 2016.

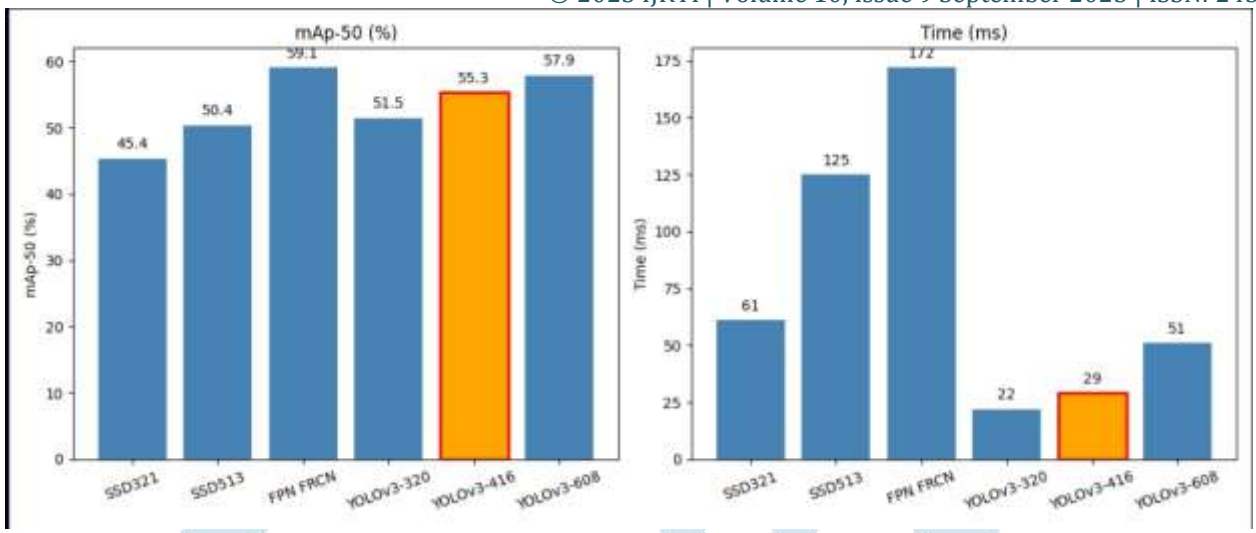


**Figure 4:** Performance and inference time of detection algorithms [13].

Figure 4 shows that while the mean average precision of the studied approaches is similar, there is a notable difference in processing durations. We chose YOLOv3-416 because of its remarkable performance, processing the COCO dataset in just 29 milliseconds, and because quick processing is crucial for our resear4ch.

## 2.2 Vehicle Tracking

Three primary steps make up car tracking, which is based on object-tracking techniques: A starting set of detected objects is taken; the detected automobiles are registered and assigned unique identifications (IDs); they are tracked as they move across a continuous frame, the ID assignments are kept, and they are erased when the cars vanish. Additionally, by counting distinct object IDs, object tracking enables us to count many autos in a frame. An object tracking algorithm should ideally meet the requirements listed below:

- The object detection step should only need to be carried out once.
- Function considerably more quickly than the real object detector.
- Respond to circumstances in which a tracked object "disappears."
- Be able to withstand occlusions.
- Identify "lost" things between frames.

### 2.2.1 Contour-Based Tracking

Determining a vehicle's boundaries is the fundamental idea of contour-based tracking. A technique for vehicle pose detection utilising optical flow and camera settings was presented by Ambardekar et al. [14]. They clarified that a part of the system estimates the three-dimensional (3D) world coordinates of the vehicle's spots in order to track the blobs and fix mistakes from the foreground object identification module. The individual cars are then segmented and tracked by grouping these points. The authors emphasised that occlusion detection and simultaneous tracking of several vehicles are features of their methodology [15].

2.2.2 Kalman Filter for Tracking

A sequence of measurements taken over time (frame by frame in our example) is called Kalman filtering, also referred to as linear quadratic estimation (LQE). A strong tracking technique for capturing motion from video sequence surfaces across point collections is the Kalman filter. The parameters of interest are inferred from indirect, imprecise, and uncertain observations using this optimum estimator. The prediction and update phases make up the Kalman filter. The first stage predicts the present state based on past states. The status is corrected in the second phase by using the current measurement, such as determining the location of the bounding box. The following significant characteristics of the Kalman filter are advantageous for tracking:

- Predict the object's future location
- adjust the forecast in light of fresh measurements;
- lessen noise brought on by imprecise detection
- Make it easier to link several objects to their tracks.

One advantage of the Kalman filter tracker over other tracking algorithms is its speed. As a result, we modified the original algorithm and made it the primary tracking mechanism in our project. Despite the wide range of approaches, our main goals are to increase detection speed, accuracy, and robustness in various illumination scenarios. As a result, we integrated the Kalman filter for tracking estimates with the YOLOv3 approach for detection.

## 3. Proposed System

### 3.1 Detection and Tracking

The three primary steps of our task are direction checking, tracking, and detection. YOLOv3 is an object detection technique designed for analysing videos in real time. The YOLOv3 model's output retrieves a vehicle's bounding box data so that reliable tracking methods can be used to determine direction.
In order to obtain bounding boxes of cars, we first submit a continuous frame to the YOLOv3 model, which has already been trained. Centroid tracking, which is a multi-step procedure, forms the foundation of our approach. For each detected object in each frame, centroid tracking assumes that we are passing (x, y)-coordinates in a set of bounding boxes. After obtaining the bounding box coordinates, we calculate the bounding boxes' "centroid," or centre (x, y) coordinates. We give them distinct IDs because this is the first set of bounding boxes that our algorithm is shown. Next, we determine the difference in cost between the predicted and detected cars. We must designate a tracker for each detection if there are several. Our metric is the intersection over union (IOU) of the detection bounding box and the tracker bounding box. The Hungarian procedure is used to maximise the sum of the IOU assignment [16]. The Hungarian algorithm is implemented by a built-in utility function in the scikit-learn machine learning package. Then, in order to forecast numerous automobile objects from a video frame, we apply updated Kalman filter equations as presented by Welch and Bishop [17].

---

**Prediction Phase:**

- **Notations:**
  - xx: State mean
  - PP: State covariance
  - FF: State transition matrix
  - QQ: Process covariance

- BB: Control function (matrix)
- uu: Control input

1. Predicted (a priori) state estimate:

   $$x=Fx+B\cdot u \quad \text{(Equation 1)}$$

2. Predicted (a priori) error covariance:

   $$P=FPF^T+Q \quad \text{(Equation 2)}$$

---

**Update Phase:**

- **Notations:**
  - HH: Measurement function (matrix)
  - zz: Measurement
  - RR: Noise covariance
  - yy: Residual
  - KK: Kalman gain

3. Innovation or measurement pre-fit residual:

   $$y=z-Hx \quad \text{(Equation 3)}$$

4. Innovation (or pre-fit residual) covariance:

   $$S_k=HPH^T+R \quad \text{(Equation 4)}$$

5. Optimal Kalman gain:

   $$K=P\cdot H^T\cdot S_k^{-1} \quad \text{(Equation 5)}$$

6. Updated (a posteriori) state estimate:

   $$x=x+K\cdot y \quad \text{(Equation 6)}$$

7. Updated (a posteriori) estimate covariance:

   $$P=(I-KH)P \quad \text{(Equation 7)}$$

8. Measurement post-fit residual:

   $$y=z-H\cdot \hat{x} \quad \text{(Equation 8)}$$

For the measurement update pair, the updated (a posteriori) state estimation formula is always valid. To project or forecast the new (a priori) estimates, the procedure is repeated using the earlier (a posteriori) estimations. method 1 is an implementation of the modified Kalman filter method. Next, we drew tracking lines on each identified vehicle using OpenCV drawing libraries.

**Algorithm 1:** Kalman Filter Tracker (Object Tracking based on Centroid Coordinates)

```
Function: ObjectTracking(x, y centroid)
for each iteration (i ≥ number of detected centroids):
   if no track vectors are found:
      Create new tracks
   Calculate the cost using the sum of squared distances
   Assign detected measurements to predicted tracks using the Hungarian algorithm
   if some tracks are not assigned:
      Identify unassigned tracks
   if tracks have not been detected for an extended period:
      Remove untracked objects
   else:
      Start new tracks for untracked objects
   Update the Kalman Filter state, store last results, and trace the tracks
```

3.2 Direction Estimation

The vehicle trajectory estimate method is explained in this section. We compute the difference between the pixels in the FN-frames range, where N- is the number of frames needed to monitor a specific vehicle, in order to determine the direction of a tracked vehicle. First, we get the centroid from the bounding box of a detected car and set it to the frame's starting positions (X0, Y0) (F0). The centroid then shifts from frame to frame as the vehicle moves. For each frame, we therefore compute the pixel difference between (X1, Y1) and (XN, YN), where X1, Y1 denotes the centroid position of the automobile in the frame (F1) and XN and YN in the frame (FN). Let's take an example where we track using six frames. The difference in pixel positions would be $\Delta X = X5 - X1$. In essence, the second frame serves as the starting point, and the sixth frame serves as the conclusion. This is because, as soon as an automobile is recognised, the ID assignment phase is carried out on the first frame. We use basic maths to determine the direction: the car is driving from right to left if X is negative and X5 is less than X1, and from left to right if X is negative. The pseudo-code used to verify the car's direction is described in **Algorithm 2.**

**Algorithm 2.** Pseudo-code for the direction check.

```
if (length of centroids > 0) {
   for (each centroid in detected centroids) {
      for (each tracking frame N for unique cars) {
         take X1, Y1 positions from the tracker
         take the unique car's ID
         ΔX = XN − X1

         if (check the sign of ΔX and compare X1 with X2) {
            assign the direction
         }
      }
   }
}
```

The shifting position of the car within N = 6 frames is illustrated graphically in Figure 5. The car's displacement through each frame is indicated by green dots. In this instance, the text "West" that is presented indicates that the vehicle in the video is travelling from right to left.

**Figure 5.** Car displacement across N = 6 frames.

3.3 Direction Validation

We developed the entry-exit method to ascertain whether the vehicle is travelling in the correct direction. Its concept is straightforward and efficient. On the video frame, we draw fictitious entrance lines. When an automobile crosses the entry lines, its unique identification number is recorded in a list; when it exits the region, the ID is deleted.

We crop the video frame to the desired region of interest, as seen in Figure 6. Lines 1–4 require traffic to move from the right side to the left. The proper entry and exit routes are indicated by green lines. Vehicles are not allowed to travel in the directions indicated by red lines (lines 5–7). We designate three hypothetical entry and departure points. Consider the typical scenario in which the vehicle pulls into the first location. In this scenario, the car's ID is saved in list 1 and stays there till the car is present from any exit after its centroid passes through region A. The car must exit in area C if it enters from area B, as the ID is recorded in list 2. If it enters from area B, the car will be identified as going in the incorrect direction and a warning signal will be activated. The car's ID is kept in list 3 to verify the region from which it left, and if the car enters area C, it is always the incorrect way. The alert signal is transmitted while the vehicle is driving, even if it enters from area C. The technique verifies based on the centroid position of the vehicle.

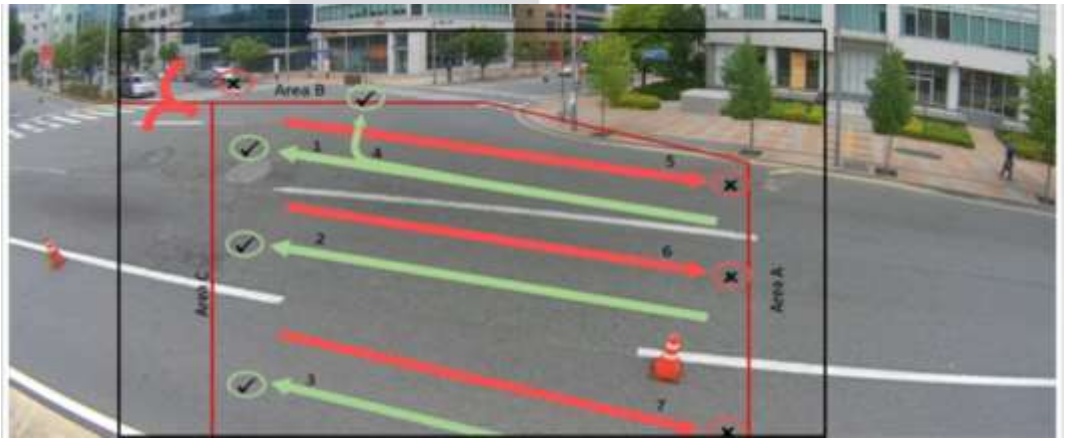


**Figure 6.** The entry-exit approach (black lines represent the borders of the frame).

## 4. Experiments and Results

To assess the effectiveness of our effort, we carried out two different kinds of trials. The first involved testing the accuracy of car detection in various illumination scenarios, while the second involved detecting drivers driving in the wrong direction. A real-world camera was made available to us, and we used it to build a simulation in which a car was travelling in the incorrect direction.

### 4.1 Datasets

A stationary CCTV camera provided the data for our study. We construct two kinds of datasets: one for improper direction checking and another for network training and testing. Twelve movies that were shot at various times of the day have been extracted for the first dataset. Every video's frames are extracted, and they are then divided at random into 70% training and 30% testing. Only data of an automobile travelling in the wrong direction was included in the second type of dataset, which is utilised to verify wrong-direction detection.

### 4.2 YOLOv3 Model Training

We annotated real-world data to train the model for the detection trials. We produced our own CCTV dataset by extracting frames from video files. On a PC with an NVidia RTX 2080Ti GPU, an Intel Core i5-7600 CPU, and 16 GB RAM, we trained our YOLOv3 network with 1750 extracted photos that we tagged using a particular software [18]. We selected YOLOv3-416, which resizes photos to 416 × 416 pixels over 100 epochs with a learning rate of $\alpha = 10\text{-}3$ and a batch size of 64 for frozen layers and 8 for unfrozen ones. With $\alpha = 10\text{-}3$ for frozen layers and $\alpha = 10\text{-}4$ for unfrozen layers, we employed the Adam optimiser. The amount of training data determines how long it takes the YOLOv3 network to converge, and in our case, it took four to five hours. Our model only works with that specific camera because it was trained on photos from the camera in a single, fixed position.

### 4.3 Detection Testing

We generated a second dataset of unseen videos to evaluate detection performance. The number of cars that appeared in those films was then manually determined. Next, we chose videos from various time periods at random. We selected morning, midday, dusk, and evening timings, among other time intervals. We choose to compute the mean average precision (mAP or μ) with IOU threshold $\mu \geq 0.75$ in order to assess detection precision. The mAP with threshold is determined by,

$$\mu = \frac{\text{bounding box area of detected car}}{\text{bounding box area from manual annotation}} \geq 0.75$$

To ascertain the automobiles' whereabouts in a frame, we manually tallied the number of cars that showed up at the specified intervals and annotated them. Only cars that were unique were counted. Only when the IOU rate was greater than 0.75 was the car deemed to have been appropriately spotted. The detection method found autos with an average mAP of 90.80% throughout the day and at night, as shown in Table 1. The varying lighting conditions are the cause of the minor variation in mAP. We evaluated our application for accuracy and speed because it is intended to be used in real-world situations. Thus, in addition to precision, we also examined our algorithm's processing time, which was comparable to Redmon and Farhadi's [16]. We

outperformed the authors of the original YOLOv3 in terms of mAP score, even though we only had one class to identify (see Figure 4).

**Table 1.** Detection network evaluation table.

| Time Interval | 10:23–10:25 | 13:49–13:52 | 15:50–15:54 | 18:05–18:07 | 21:10–21:13 | Results |
|---|---|---|---|---|---|---|
| Num. of cars | 30 | 44 | 85 | 50 | 41 | 250 |
| Detected cars | 27 | 40 | 80 | 47 | 40 | 234 |
| mAP > 0.75 (%) | 90.00 | 93.18 | 94.12 | 88.00 | 85.36 | 90.80 |
| APT (ms) | 27 | 25 | 26 | 28 | 29 | 27 |

mAP: mean average precision; APT: average processing time (ms: milliseconds).

## 4.4 Wrong-Direction Detection Testing

A qualitative experiment was presented to support the effectiveness of our efforts. evaluating for wrong-way detection is based less on evaluating the preexisting assumption and more on the structure of the relevant data. Thus, we used both real and simulated data to test our implemented "entry-exit" algorithm for wrong-direction identification. Because real wrong-way driving occurrences are hard to obtain, we simulated them by purposefully driving a car in the wrong direction. We examined daylight wrong-direction detection in the first simulation scenario. A car entering from area C in the wrong direction is monitored until it reaches area A, which is where it exits in our scenario, as shown in Figure 5. Images and details about the vehicle were transmitted to the monitoring centre via the Transmission Control Protocol (TCP) and File Transfer Protocol (FTP) as soon as it arrived in area C. An 807 × 646 video frame at 30 frames per second is used by the algorithm to identify moving cars.

The Korean Road Traffic Authority (KoROAD) forbade us from simulating driving at night for safety reasons. In order to verify true-positive results, we reversed the daytime and nighttime videos for the second and third simulation situations, as illustrated in Figure 7. Table 2 displays the results of the first simulation scenario, where "positive direction" refers to the number of cars travelling in the right direction and "negative direction" refers to the number of incidents of cars driving in the incorrect direction. In case 1, where N is the total number of distinct autos, we looked at the scenario where the vehicle was driven incorrectly during the day.

We looked at a horizontally flipped video from case 1 for case 2, and we used a flipped nighttime video for case 3 to verify the correctness at night. We reversed the video frame because Monteiro et al. [8] employed the image flipping technique to verify the precision of their system. We can gauge the effectiveness of our wrong-direction detecting algorithm (Table 3) using Table 2. Figure 7 displays the visual results.

**Table 2.** Confusion matrix for three simulation cases (quantity of cars).

| Case 1: N = 114 | Predicted Positive Direction | Predicted Negative Direction |
|---|---|---|
| Positive direction | 100 | 0 |
| Negative direction | 0 | 6 |
| Case 2: N = 114 | | |
| Positive direction | 0 | 0 |
| Negative direction | 15 | 105 |
| Case 3: N = 36 | | |
| Positive direction | 0 | 0 |
| Negative direction | 7 | 33 |

**Table 3.** Performance measurement in (%).

| Cases | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Case 1: Daytime | 100.00 | 100.00 | 100.00 | 100.00 |
| Case 2: Daytime (flipped) | 84.83 | 100.00 | 84.00 | 90.44 |
| Case 3: Nighttime (flipped) | 83.11 | 100.00 | 81.00 | 93.53 |

False-positive results were found by another evaluation we performed. To determine the quantity of false-positives, the system was put through a 24-hour test. Only one false-positive was found by our algorithm, which counted 5551 automobiles that passed by the camera in a 24-hour period. This was because the car was illuminated at night (Table 4).

**Table 4.** Confusion matrix for real-world cases.

| N = 5552 | Predicted 'NO' | Predicted 'YES' |
|---|---|---|
| Actual 'NO' | 5551 | 1 |
| Actual 'YES' | 0 | 0 |

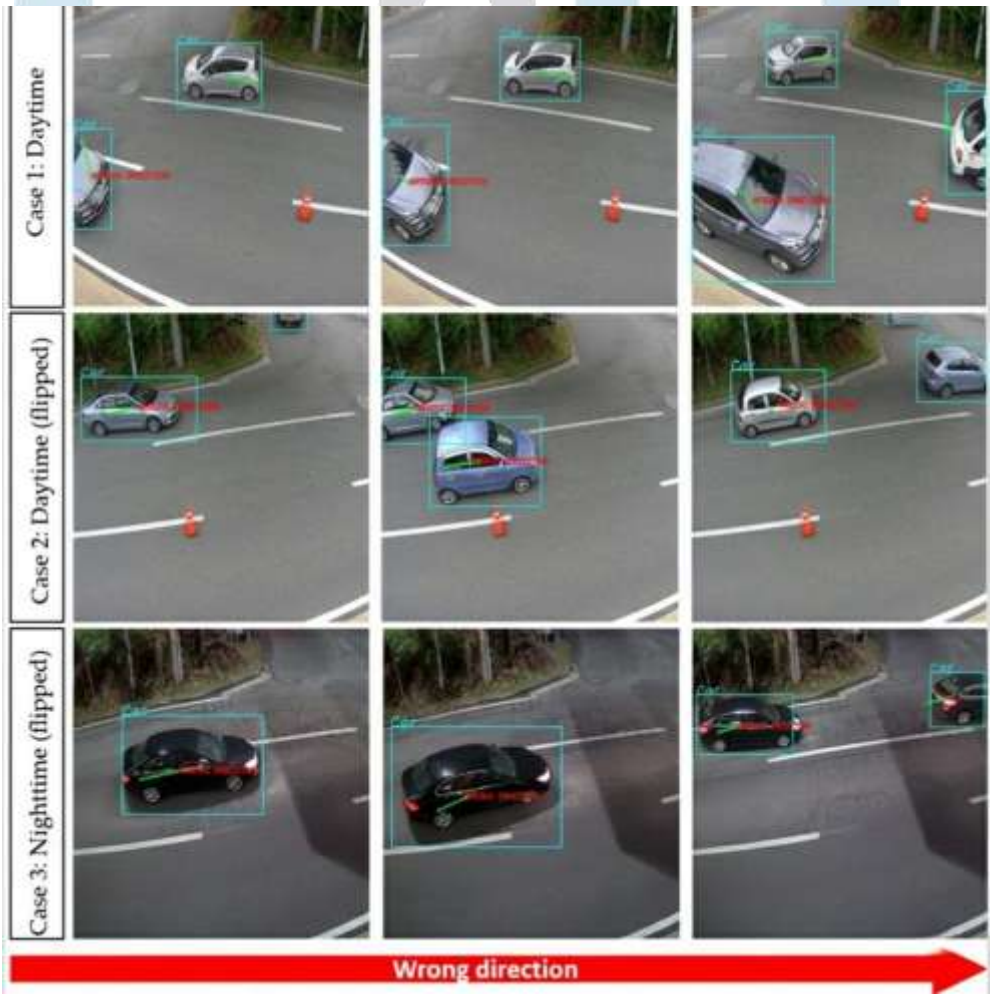


**Figure 7.** Wrong-direction detection for the three simulation cases.

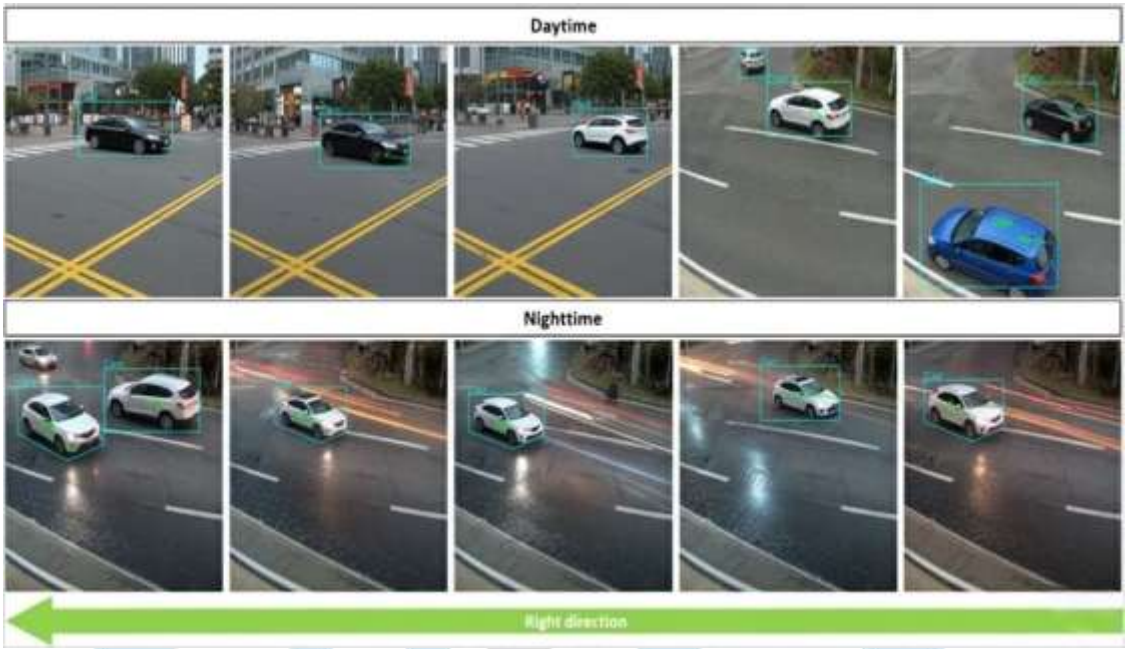


**Figure 8.** Real-time detection example during both daytime and nighttime.

## 5. Conclusions and Future Work

To detect and track cars, we suggest combining the most recent YOLOv3 technique with an updated linear quadratic equation. We developed a novel "entry-exit" technique to detect vehicles moving in the wrong way, and we were able to simulate the three scenarios and obtain an averaged accuracy of 91.98% in the wrong-direction identification findings. This system, as demonstrated by the simulation results, shows outstanding performance in a range of illumination conditions, allowing us to use real-world cameras to run the program. As part of ITS solutions, the present system, which operates at 30 frames per second, was installed on one of the South Korean roadways. Notwithstanding the strong performance, our next goal is to expand our network's detecting capabilities. The next phase of our research is to include violation detection of jaywalking since we discovered numerous instances of people crossing the road in a restricted location throughout our experiments.

**References**

1. Tzutalin. LabelImg. Git Code. 2015. Available online: https://github.com/tzutalin/labelImg (accessed on 30 January 2020).
2. Kanhere, N.K.; Birchfield, S.T.; Sarasua, W.A. Vehicle Segmentation and Tracking in the Presence of Occlusions. *Transp. Res. Board Annu. Meet.* **2006**, *1944*, 89–97. [CrossRef]
3. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767v1.
4. Ambardekar, A.; Nicolescu, M.; Bebis, G. Efficient vehicle tracking and classification for an automated traffic surveillance system. *Signal Image Process.* **2008**, *623*, 101.

5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection.
   *arXiv* **2015**, arXiv:1506.02640.
6. Bouwmans, T.; Baf, F.E.; Vachon, B. *Background Modeling Using Mixture of Gaussians for Foreground Detection—A*
   *Survey*; Recent Patents on Computer Science; Bentham Science Publishers: Shaka, UAE, 2008; Volume 1, pp. 219–237.
7. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector.
   *arXiv* **2016**, arXiv:1512.02325.
8. Barron, J.L.; Fleet, D.J.; Beauchemin, S. Performance of optical flow techniques. *Int. J. Comput. Vis.* **1994**, *12*, 43–77. [CrossRef]
9. Shi, J.; Tomasi, C. Good features to track. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June 1994; pp. 593–600.
10. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. *Proc. Imaging Underst. Workshop* **1981**, *1981*, 121–130.
11. Zivkovic, Z. Improved adaptive gaussian mixture model for background subtraction. In Proceedings of the 17th International Conference on Pattern Recognition 2004. ICPR 2004, Cambridge, UK, 26–26 August 2004; Volume 2, pp. 28–31.
12. Zivkovic, Z.; Van Der Heijden, F. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognit. Lett.* **2006**, *27*, 773–780. [CrossRef]
13. Guo, G.; Wang, H.; Bell, D.A.; Bi, Y.; Greer, K. KNN Model-Based Approach in Classification. In *Lecture Notes in Computer Science, Proceedings of the OTM: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Sicily, Italy, 3–7 November 2003*; Springer: Berlin/Heidelberg, Germany, 2003. [CrossRef]
14. Kuhn, H.W. The Hungarian Method for the assignment problem. *Naval Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]