


Article

Design and Implementation of a Highly Scalable, Low-Cost Distributed Traffic Violation Enforcement System in Phuket, Thailand

Somphop Limsoonthrakul ^{1,2,*} , Matthew N. Dailey ^{2,3}, Ramesh Marikhu ^{2,3}, Vasan Timtong ², Aphinya Chairat ², Anant Suphavitai ⁴, Wiwat Seetamanotch ⁵ and Mongkol Ekpanyapong ^{1,2}

¹ Industrial Systems Engineering, School of Engineering and Technology, Asian Institute of Technology, Pathumthani 12120, Thailand; mongkol@ait.asia

² AI Center, Asian Institute of Technology, Pathumthani 12120, Thailand; mdailey@ait.asia (M.N.D.); st115521@ait.asia (R.M.); vasant@ait.asia (V.T.); aphinya@ait.asia (A.C.)

³ Information and Communications Technologies, School of Engineering and Technology, Asian Institute of Technology, Pathumthani 12120, Thailand

⁴ Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK; as1y19@soton.ac.uk

⁵ Phuket Provincial Public Health Office, Phuket 83000, Thailand; swiwat2@yahoo.com

* Correspondence: st11486@ait.asia

Abstract: The number of global road traffic accidents is rising every year and remains undesirably high. One of the main reasons for this trend is that, in many countries, road users violate road safety regulations and traffic laws. Despite improvements in road safety legislation, enforcement is still a major challenge in low- and middle-income countries. Information technology solutions have emerged for automated traffic enforcement systems in the last decade. They have been tested on a small scale, but until now, the cost of deployment of these systems is generally too high for nation-wide adoption in low- and middle-income countries that need them the most. We present the architectural design of a traffic violation enforcement system that can optimize the cost of deployment and resource utilization. Based on the proposed architecture, we describe the implementation and deployment of the system, and perform a comparison of two different versions of the video-based enforcement system, one using classical computer vision methods and another using deep learning techniques. Finally, we analyze the impact of the system deployed in Phuket, Thailand from 2017 to the present in terms of local road users' compliance and the road safety situation. We conclude that the system has had a positive impact on road safety in Phuket at a moderate cost.

Keywords: traffic violation enforcement system; computer vision; deep learning; road safety



Citation: Limsoonthrakul, S.; Dailey, M.N.; Marikhu, R.; Timtong, V.; Chairat, A.; Suphavitai, A.; Seetamanotch, W.; Ekpanyapong, M. Design and Implementation of a Highly Scalable, Low-Cost Distributed Traffic Violation Enforcement System in Phuket, Thailand. *Sustainability* **2021**, *13*, 1210. <https://doi.org/10.3390/su13031210>

Received: 21 December 2020

Accepted: 21 January 2021

Published: 24 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

According to the global status report on road safety published by the World Health Organization (WHO) in 2018 [1], the number of road traffic accidents and fatalities is increasing every year, and road accidents are the primary cause of death for children and young adults. The situation is more severe in low- and middle-income countries, where the level of compliance with traffic laws is not keeping up with automobile performance and transportation systems. In attempts to meet WHO goals for the reduction of traffic accidents, most governments are tackling key risk factors for road safety, such as over-speeding and helmet wearing, by establishing social marketing campaigns, strengthening legislation, and intensifying law enforcement. While these strategies must be established together for the most effective improvement of road users' behavior, robust law enforcement would have the most immediate impact in countries where law enforcement has been deficient. As an example, we can consider differences in motorcycle helmet law compliance in Vietnam and Thailand. The two countries share a similar culture in which motorcycles

are a primary form of transportation. Despite having strict helmet laws in both countries, Vietnam has a much higher helmet-wearing rate than Thailand; 81% of drivers and 60% of passengers wear helmets in Vietnam, while 50% of drivers and 20% of passengers wear helmets in Thailand. The reason is that helmet law enforcement in Vietnam is more intense than in Thailand—enforcement levels are 8 and 6, respectively, on a scale of 0 to 10. A similar situation is occurring for speed law enforcement in the two countries. The WHO report emphasizes that the intensity of traffic law enforcement in many countries is generally weak for example, helmet law enforcement in 114 out of 175 participating countries is rated below 8. Most of these countries are developing countries in which the number of officers and the budget allocated for traffic law enforcement are insufficient compared to the number of violations occurring every day.

In recent decades, many countries have adopted automatic systems for transportation monitoring and traffic law enforcement. Relying on computer systems and many kinds of sensors, these technologies are improving transportation management and road safety [2–4]. Some systems rely on traditional sensors installed on road surfaces, such as magnetometers [5–7] and loop detectors [8,9], which enable the use of a simple system architecture with processing computers operating locally. With improved sensory technologies and computation technologies, some systems adopt more advanced sensors, such as radar [10,11], LIDAR [12–14], and computer vision techniques with CCTV cameras [15–28]. However, although highly capable, these systems require costly infrastructure that is difficult for low- and middle-income countries to deploy extensively. In more recent years, technology has moved toward more cost-effective distributed systems that rely on embedded systems installed locally on vehicles and vehicular ad-hoc networks (VANETs) [29–33]. These systems are also difficult to apply in low- and middle-income countries because it is nearly impossible to force every road user to equip their vehicles with the needed embedded systems. Moreover, security and privacy are still questionable for such technologies [34–36].

Among all methodologies mentioned previously, the use of computer vision techniques with CCTV cameras installed on the roadway infrastructure is best suited to traffic violation enforcement systems in developing countries due to the low cost of hardware and ease of maintenance. However, there are two main concerns that must be taken into consideration when deploying the system over a large physically distributed area. First is the computing resources required to process rich visual information that comes from video cameras, and second is the type of network infrastructure required to transmit high-bandwidth video streams from the cameras to the computing resources. As the number of deployments and the area of coverage increase, the budget required for system installation can increase dramatically. Hence, the design of the system architecture is critical to reducing the cost of initial installation and allowing the system to scale up with modest incremental investments.

In this article, we present the design and implementation of a traffic violation enforcement system that optimizes resource utilization and cost of deployment. The system platform is designed as a roadside system, which is more practical and less privacy-invasive than in-vehicle systems. We use CCTV cameras for the effective and cost-efficient detection of traffic violations. Our system has been deployed in Phuket, Thailand since 2017 under a road safety improvement project funded by the Safer Roads Foundation (SRF), and it was scaled up during a follow-on project in 2018 funded by Thailand's National Electronics and Computer Technology Center (NECTEC). We chose intersections and junctions as target locations for our deployments, as a study by a local agency found that most urban traffic accidents occur close to intersections or junctions (this is true even in developed countries, as reported by the US Department of Transportation [37]). We develop vision-based detection of multiple traffic violations that frequently occur at intersections including red-light running, motorcycling without a helmet, and over-speeding. Our main contributions are as follows:

- We present a distributed modular software architecture, the Distributed Road Safety System Architecture (DRSSA), which can reduce the cost of network infrastructure and introduce robustness, flexibility, and scalability into the system;
- We present the implementation of video analytics to detect three types of traffic violations: Red-light running, motorcycling without a helmet, and over-speeding;
- We compare the accuracy of two versions of each of the video analytics, one based on classical computer vision techniques and one based on deep learning techniques, and we discuss the cost efficiency and scalability of DRSSA, as assessed over two separate deployment phases;
- We analyze the impact of the enforcement system on the number of violations and accidents, local driver compliance, and road safety in Phuket, Thailand where our proposed system has been deployed since 2017.

The rest of this paper is organized as follows. Section 2 gives an overview of existing work on vision-based violation detection systems. In Section 3, we present an overview of the Distributed Road Safety System Architecture (DRSSA) and the structural design of each component in the system. Then, we give implementation details of both versions of our video analytics in Section 4. The results obtained thus far from the system deployed in Phuket and its impact on local road safety are discussed in Sections 5 and 6, respectively, followed by a conclusion in Section 7.

2. Related Works

The technologies for automated traffic violation detection have been evolving over decades. CCTV cameras are the most popular sensors, since they not only provide a cost-effective solution, but also offer additional functionalities such as traffic monitoring and surveillance. Despite the benefit of rich visual information provided by video cameras, the accuracy of vision-based traffic violation detection can be strongly affected by occlusion, illumination, shadows, and weather conditions. Hence, a number of innovative techniques have been proposed to provide robust and accurate video analytic algorithms in traffic violation detection systems.

2.1. Classical Computer Vision Methods for Traffic Violation Detection

As described by Loce et al. [38], a basic computation pipeline for vision-based transportation applications is shown in Figure 1. The pipeline starts with video data retrieved from one or more cameras capturing traffic scenes at deployed locations. The visual data are passed to the next stage, called pre-processing, which extracts image sequences out of the video stream and performs image enhancement operations such as brightness and contrast correction, distortion compensation, and noise filtering. This step also includes some preliminary tasks before processing the image in the next step such as resizing, cropping, and identifying regions of interest (ROIs). The third and fourth steps in the pipeline are feature extraction and decision/inference accordingly. Feature extraction is the process of transforming the raw data in pixels into higher quantitative data such as edges, histogram of oriented gradients (HOG) features [39], Haar-like features [40], and scale-invariant feature transform (SIFT) [41]. These features can be interpreted and processed by the decision operations in the next step. Together, these two steps arranged in a pipeline can define the methods for vehicle detection and violation classification (making a decision as to whether any violations that arise in the image sequences or not). In early work, vehicle detection and classification were achieved using background subtraction techniques such as histogram-based filtering [42], the single Gaussian background model [43], and the Gaussian mixture background model (GMM) [44]. Other detection and classification approaches are based on object recognition schemes. Feature points such as SIFT and speeded-up robust feature (SURF) can be used to describe the appearance of vehicles in road scenes [45,46]. However, these two types of feature descriptors require a heavy process, and processing frame rates suffer when there is large number of vehicles in the scene (especially at intersections). The use of support vector machines (SVMs) for HOG feature classification and the use

of online adaptive boosting (AdaBoost) with Haar-like features enable a more accurate detection of vehicles in real time [47,48]. These same two techniques have also been applied to vehicle type classification [49,50].

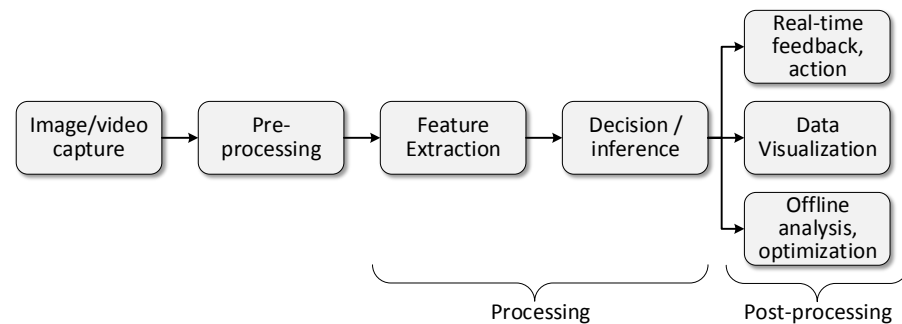


Figure 1. Computer vision pipeline for transportation applications.

In our first deployment of traffic violation detection, we adopted two main approaches, HOG features + SVM and Haar-like features + Adaboost, for both vehicle detection and object classification. Details of our implementation are provided in Section 4.

2.2. Deep Learning Methods for Traffic Violation Detection

Over recent years, as computational technologies such as parallel computing on GPUs have been gradually accelerated, neural network or deep learning methods have come to dominate tasks in computer vision. The convolutional neural network (CNN) is a kind of deep neural network architecture designed to process spatial information in digital images. In the context of vision-based transportation applications, CNNs have been adopted in two domains: Detection and classification. The very first CNN architectures were designed to solve problems of image classification. AlexNet [51] was proposed in 2012. The network architecture is an extension of the classical LeNet classifier [52] with a larger number of layers and more filters per layer. The network structure is split into two sub-networks in order to benefit from parallel processing on multiple GPUs, and this model immediately became one of the most popular networks for classification. Another two commonly used neural-network classifiers are GoogLeNet [53] and VGGNet [54], which were the winner and runner-up, respectively, at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 [55]. With the introduction of “inception” modules, GoogLeNet achieved deeper network structures while avoiding problems of overfitting and unnecessarily expensive computation. The network architecture of VGGNet is similar to AlexNet but has many more filters in the convolution layers. VGGNet has a very uniform architecture but is harder to train due to a large number of parameters. Another famous classifier is the Residual Neural Network (ResNet) [56] which won ILSVRC in 2015. ResNet introduced so-called “skip connections” and make effective use of heavy batch normalization, which enable the architecture of ResNet to have more layers but lower complexity than VGGNet.

CNNs for object detection are much more complex than CNNs for classification. The main reason is that the length of the output varies with the number of objects in the image. A simple solution to this issue, commonly used by classical computer vision methods, is to process on different regions of the image. However, this method is very computationally expensive and its accuracy relies on how densely or intelligently the regions are sampled. To solve the problem of selecting too large a number of regions, R-CNN [57] was proposed in 2014. It uses selective search to generate candidate region proposals. However, even with selective search, R-CNN still takes a long time to process a single image, so it cannot be used for real-time detection. Two improved versions of R-CNN, called Fast R-CNN [58] and Faster R-CNN [59], were proposed in 2015 and 2016.

Fast R-CNN replaces the pre-process of selective search on the input image before feeding each region proposal to the network with a selective search on a convolutional feature map inside the network, which only requires the image to be fed to the network once. However, Fast R-CNN is still too slow for real-time detection. Faster R-CNN completely removes the selective search algorithm and instead allows the network to learn to find region proposals. Faster R-CNN achieves much lower inference time than other methods and can be used for real-time applications. Another state-of-the-art neural network detector is called You Only Look Once (YOLO) [60]. YOLO is different from the region-based algorithm in the R-CNN series in that region proposal finding and object classification are performed sequentially in these methods. YOLO performs both tasks in parallel. The input image is split into a $S \times S$ grid of cells and each cell predicts B candidate bounding boxes along with the probability of an instance of each object class being contained inside each bounding box. YOLO outperforms Faster R-CNN in terms of inference speed despite a slight drop in detection accuracy.

In our second deployment of DRSSA, we upgraded our processing units with GPUs suitable for deep learning inference. Moreover, we improved our detection and classification algorithms by replacing classical image processing methods with more modern deep learning approaches. We adopted YOLO for the vehicle detector and GoogLeNet for the classifier. Details of implementation are provided in Section 4.

3. System Architecture

A large-scale traffic enforcement system contains many deployment locations and must be able to handle a large number of concurrent users. In this situation, there are many development aspects that must be considered, e.g., the project's budget, system maintainability, and scalability. In this paper, we describe the design of Distributed Road Safety System Architecture (DRSSA), an architecture for a traffic-violation enforcement system that can be implemented over a citywide area with low cost and has high availability, maintainability, and scalability. The system is organized as a layered architecture—components are organized into layers according to their role. As shown in Figure 2, the system architecture consists of four layers: Edge, processor, application, and user. Each layer of the architecture has a specific responsibility as defined by its name and only interacts with adjacent layers. The implementation detail of the system components in each layer are described in the following sections.

3.1. Edge Layer

The first layer, the edge layer, is where violation detection sensors, such as IP cameras and speed guns, are located. These sensors are deployed at various locations where the police officers would like to observe violations and enforce traffic laws. For each location, the number of required sensors depends on types of violations and the observation area (one traffic lane or multiple traffic lanes). On a particular site, we set up a local area network linking the sensors and analytic machines running in the processor layer. This local network requires a high data bandwidth so that the HD video streams and speed sensor data can be transmitted and processed by analytic software in real time. Considering a typical large number of data streams and the typical long distance from target location to the relevant police station, we normally use optical fiber for the local sensor network.

A representative configuration of sensors in our system implementation is shown in Figure 3. The violation detection system normally requires $n + 1$ cameras, where n is the number of traffic lanes. One camera is typically setup to capture an overview of the scene (covered by yellow shaded area), and the remaining n cameras are equipped with telephoto lenses to take clear snapshots of traffic violators' license plates (focusing on orange shaded areas). When an over-speeding violation is targeted at the location, additional measurement sensors such as speed guns can be installed alongside the cameras. All sensors are typically mounted on a single camera pole above the road in order to avoid occlusion of target vehicles. The cameras must point in the same direction as the traffic

flow in order to capture the vehicle from the rear. Setting up the sensors in this way allow the capture of license plates of any kind of violating vehicles including motorcycles, which only have rear-mounted license plates in Thailand.

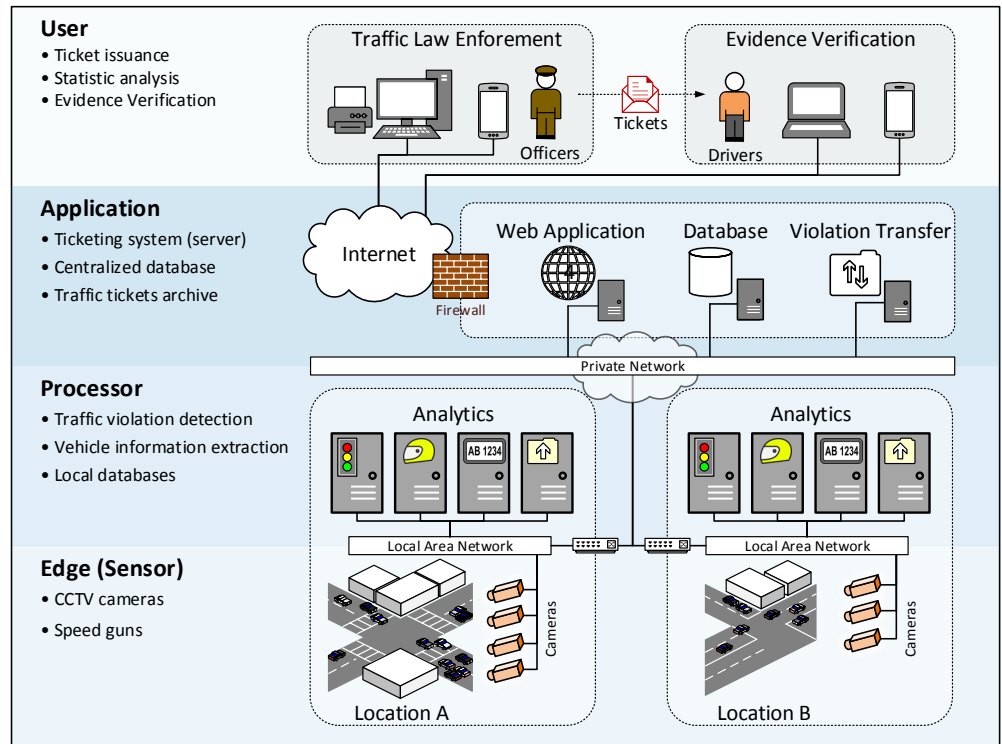


Figure 2. Overview of Distributed Road Safety System Architecture (DRSSA).

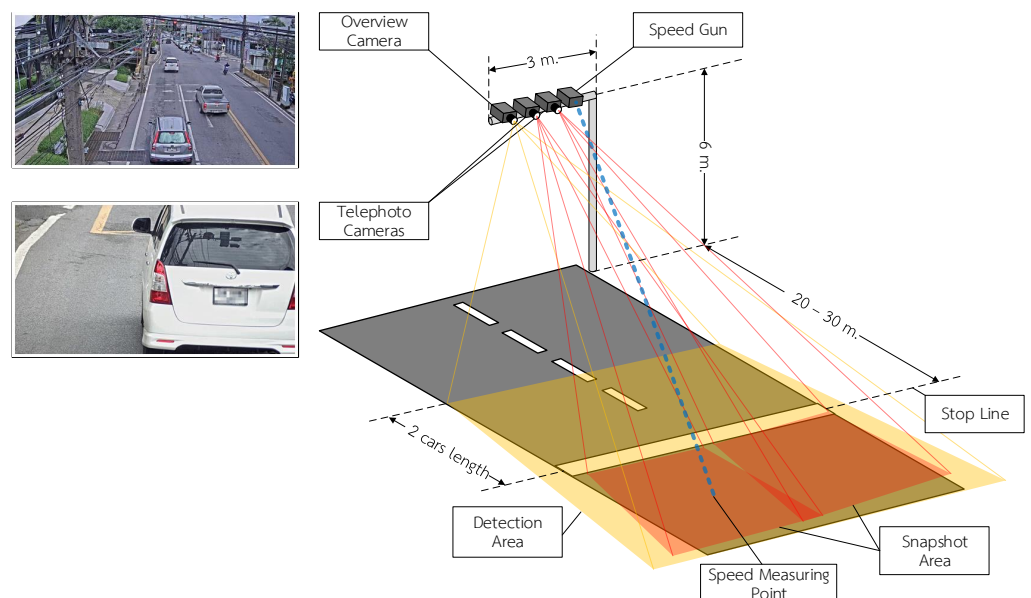


Figure 3. Sensor configuration in DRSSA.

3.2. Processor Layer

The processor layer contains video analytic software that processes sensor data arriving from the edge layer, and records evidence of any violations found. There are four main analytic processors in the system: *Red-light violation detection*, *helmet violation detection*,

speeding violation detection, and *vehicle information extraction*. Implementation details of each analytic are given in Section 4. As was mentioned in previous sections, the processor layer is designed for distributed computation, co-located with the sensors, instead of running on a single centralized processor. The main reason for this architecture is to reduce the cost of installation. Since the system is installed over a city-wide area and since each location is distant from the others, having a high-bandwidth network tightly linking every node in the system will increase the installation cost substantially. In order to make the system affordable in middle income countries such as Thailand, we designed the system so that processors or analytic machines are distributed and installed close to the target locations, requiring a shorter range for high-bandwidth network elements. This approach also makes the system more modular because all the data is processed locally at the deployed locations and only a small amount of data is transmitted to the data center. This distributed processing approach allows connections between processors and the application server in the next layer to be more flexible (it can be via ADSL, mobile network, etc.), which increases the robustness and scalability of the system. However, all of the connections between analytic machines and application servers are through a secure virtual private network in order to ensure confidentiality.

We also define software architecture for traffic violation detection programs or processors. The purpose of this architecture is to define a modular software structure that can be easily configured, maintained, and extended. The architecture also defines a system structure that allows administrators to conveniently monitor and control the processes.

3.2.1. Analytic Module

In the DRSSA, all analytic module shares a similar process flow that can be generalized into three steps: Grabbing images from IP cameras, analyzing the images, and recording evidence data (photos, videos, and metadata) if a violation is found. The only specialized procedure in each analytic module is the second step, which defines how the program analyzes video frames and detects violations. The two other steps can be generalized into abstract sub-modules: *Image Grabber* and *Violation Recorder*. With these elements, the analytic module architecture is shown schematically in Figure 4.

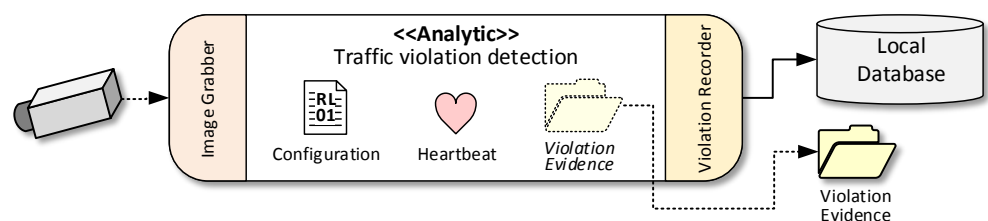


Figure 4. Architecture of analytic modules.

As shown in the figure, each analytic module consists of three main parts according to the three general processing steps described earlier. *Image Grabber* handles the connection to and retrieves video stream data from IP cameras. It decodes each stream into image frames and buffers them locally so that the analytic can process them when it is ready. The analytic processing is what each traffic violation program must define independently. However, there are three artifacts that every analytic contains:

- *Configuration*: A text file used by the system administrator to configure the behavior of the analytic;
- *Heartbeat*: A timestamp file used for checking runtime status of processes; and
- *Evidence*: A collection of files providing evidence of violations, including image files, video files, and a description text file.

The heartbeat file is created by every analytic instance, and it periodically updates the last active time of a particular analytic. This timestamp file is monitored by the *Analytic Process Manager (APM)*. To record evidence such as images and short video clips, analytics

do not need to know exact location of their file storage. They only write evidence files to a relative file path. The actual evidence directory is resolved and mapped to analytics' local directory by *APM*. Details of *Analytic Process Manager* will be described in Section 3.2.2.

Lastly, *Violation Recorder* is an interface that handles the analytic module's connection to the local database on the analytic machine. It not only manages the recording of violations in the database, but it also creates metadata for violations such as location and timestamp. With this architecture, each analytic is straightforward to implement. Developers do not need to handle complexities of camera interfaces or output structures. Moreover, the program instance can be integrated with the *Analytic Process Manager* which handles the monitoring and control of the analytic processes automatically.

3.2.2. Analytic Machine

In the previous section, we described how the architecture of analytic modules is defined so that the core analytic is easily implemented and modified. Besides modularity, the system also needs reliability and maintainability. This requires processes or program instances to be aware of faults and to be able to recover itself from failure. Hence, we introduce a common software architecture for each analytic machine so that it can increase the availability of the system and also maximize the utilization of computational resources.

Firstly, in order to implement fault detection and recovery functionality, we need a manager to monitor and control the program instances as described in Section 3.2.1. This manager is called the *Analytic Process Manager* or *APM*. See Figure 5, where the *APM* is defined by blue rounded rectangle boxes. It contains the analytic program instances and three other modules:

- *Process Monitor*: A daemon process that regularly checks the status of each analytic instance, including CPU usage, memory usage, and last active time;
- *Scheduler*: A daemon that execute specific tasks at designated times; and
- *Logs*: A message logging subsystem including information and error reports from analytic instances.

The benefit of *APM* is that we can detect faults that occur during analytic execution and prevent the system from failure. By presetting alert criteria beforehand, *APM* can check the current status of analytic instances as to whether they are behaving abnormally and reset them before they crash. The three main criteria are CPU usage, memory usage, and last active time (last time of update to the *Heartbeat* file). *APM* also checks the existence of analytic instances and revives them after an accidental crash. *Scheduler* is another module defined within *APM*. It is mainly used to restart analytic programs in different modes for example, vehicle detection models are normally scheduled to switch from day mode to night mode every evening at 6:00 p.m. Besides that, *Scheduler* is used to prevent system failure by resetting the analytic instances periodically to prevent anticipated crashes, e.g., to prevent failure due to a slow memory leak or to reset accumulated error. Lastly, *APM* incorporates a logging mechanism that, when enabled, captures the text output from every analytic instance and writes the data to log files. This helps developers and system administrators easily identify root causes of problems when any fault happens in an analytic module.

As already mentioned, DRSSA is based on a distributed processing approach. Eventually, the traffic violation output from processors or analytic programs must be transmitted and collected centrally. This occurs at the *Application Server* layer. However, this transmission can fail due to unstable connections between analytic machines and the centralized server. To prevent such data loss, we design the system architecture of each analytic machine to contain a local database that acts like a buffer for traffic violation output. The local database also simplifies the analytic module because it only needs to output to a local database. However, this does require a separate module running on the analytic machine to handle this task.

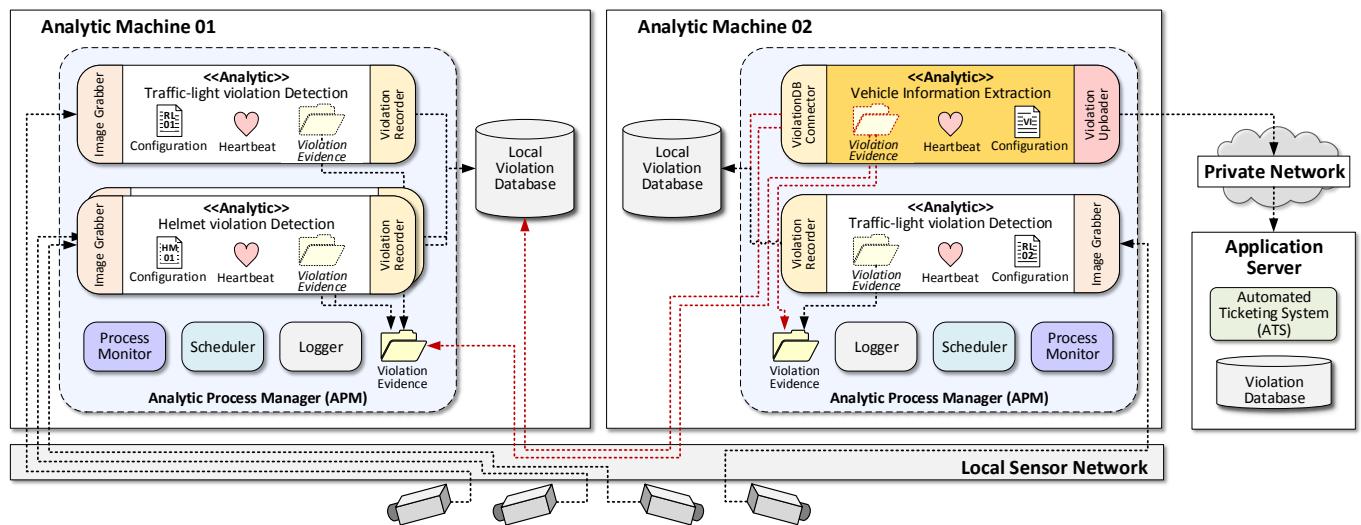


Figure 5. Overview of analytic machine structure for two machines running at one site.

We finally implement a special analytic module that is responsible for uploading violation data to the central server. This special analytic has additional functionality, especially extracting information from the violator's vehicle, i.e., vehicle type, color, brand, and license plate number. This analytic is called the *Vehicle Information Extractor* or *VInfo*, defined by the dark orange rounded rectangle. It has the same architecture as the other analytics. However, instead of taking input from sensors or network cameras, it queries and retrieves evidence data from the local database. After extracting vehicle information, it uploads the result, containing violation evidence and metadata, to the server. With the benefit of the modular structure of all analytics, *VInfo* can be configured so that every analytic machines contain one *VInfo* instance or, depending on the overall configuration, one *VInfo* instance can be shared among a group of analytic machines. Figure 5 shows an example of the second configuration. There is one *Vehicle Information Extractor*, running on *Analytic Machine 02*, operating for both *Analytic Machine 01* and *Analytic Machine 02*. The advantage of this configuration is that the available computational resources can be optimally utilized.

3.3. Application Layer

To store the results of analytic processing and to provide a tool for police officers, we designed a sub-system called the *Application* layer. This layer contains centralized components of the system including a web application server, a mobile application server, a database server, and a violation data transfer server. The web application server and mobile application server provide visualization and law enforcement tools for police officers, called the *Automated Ticketing System (ATS)*. The database server is a centralized data store comprising evidence of violations from all target locations and also records traffic tickets issued by police officers. The violation transfer server is a web-based application that provides an interface to analytics in the processor layer. It verifies the credentials of the uploading analytic, verifies the correctness of the evidence, and handles the evidence uploading process. In contrast to the analytic programs in the processor layer, all sub-systems in the application layer are centralized so that sufficient resources can be dedicated to serve a large number of concurrent users. A centralized server simplifies administration and security. It is also more cost efficient for hardware maintenance and scaling.

As shown in Figure 2, the application layer has two different network interfaces. One is a public network connected to the Internet allowing end users to access the system via web/mobile applications and another is a virtual private network connected to other components in the system. The public interface is strictly controlled by a firewall

that only allows HTTPS connections to the web application server. The end users (police officers and traffic ticket recipients) are authenticated by standard application-level username/password authentication. On the other hand, there are several services that are connected through the virtual private network, such as HTTPS for violation uploading and SSH for system administration. Processors or analytic machines can upload violation data to the centralized database via a web service provided on the server. We use X.509 client certificates to identify each uploading system. The central server acts as a SSL certificate authority (CA) for the connected analytic machines. Client secret keys and certificates are generated on the server based on basic information of the analytic machine, such as locations and analytic types. The system administrator can add, edit, and remove access for analytic machines conveniently from the control center on the centralized server.

3.4. User Layer

The last part of the system architecture, through which police officers can retrieve traffic violation evidence and issue traffic tickets via the web-based tool (*ATS*) in the application layer, can be referred to as the user layer. By using any Internet-connected device such as personal computer or mobile phone, authorized law enforcement officers can obtain a list of traffic violations in their jurisdiction and rapidly issue electronic tickets if the evidence is sufficient. The resulting citations can be printed and sent to the driver's residence by postal mail. Offenders can verify the violation evidence (video and full resolution images) online using the URL and a temporary username/password printed on the citation.

4. Implementation of Analytics

In this section, we describe the specific workflow implemented for each type of traffic violation and the algorithms adopted for video processing. We developed two different versions of each video analytic and deployed them in Phuket, Thailand. The first version of each analytic was implemented using classical image processing techniques and deployed in 2017. During a follow-on project in 2018, we improved the video analytics using more modern deep learning techniques.

4.1. Red-Light Violation Detection

Our implementation of the red-light violation detection system only uses video processing. It does not require additional sensors to determine the traffic light status or to detect vehicles. Using only the video streams from a suite of IP cameras, the red light system can detect vehicles violating the red light and can record all of the evidence required for police officers to enforce traffic law by issuing electronic citations to the violators.

According to the sensor configuration described in Section 3.1, the red-light violation detection system takes $n + 1$ video streams. One stream from an overview camera is used for vehicle detection, vehicle tracking, and recognition of traffic light status. n streams from telephoto cameras are used to take high-definition photos of a violating target vehicle and its license plate. Figure 6a shows a flowchart of the red-light violation detection system's operation. To simplify the system flow, there is only one stream from telephoto cameras shown in the diagram. However, in real scenarios, the number of telephoto cameras should be equal to the number of traffic lanes. To handle the multiple concurrent video streams, the red-light violation detection system spawns a separate process to handle each stream.

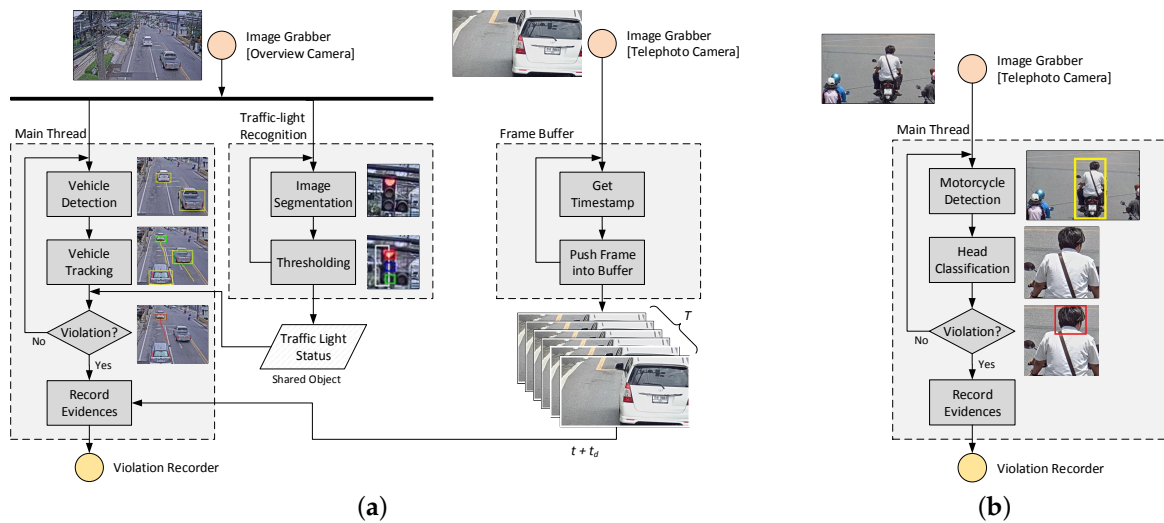


Figure 6. Flowcharts of (a) red-light violation detection and (b) helmet violation detection.

The stream from the overview camera passes through two sub-processes: *Main Thread* and *Traffic-light Recognition Thread*. As shown in the flowchart, *Main Thread* first detects all vehicles appearing in each image frame. We adopt different detection algorithms for the two versions deployed thus far:

- *Haar AdaBoost cascade classifier*: This version uses the classic AdaBoost cascade classifier using Haar-like features. We created a Haar cascade detection models from 5000 rear images of vehicle and 5000 negative images (images not containing vehicles). We trained separate modes for cars and motorcycles. Furthermore, for each detector, we created two separate models, one for daytime and another for nighttime, in order to improve the accuracy of detection during the nighttime;
- *Convolutional Neural Network*: We adopted the YOLO architecture because it can perform detection and classification in real time (45 FPS with a mean average precision (mAP) 63.4%). We train the YOLO network from scratch with two different output classes: Cars and motorcycles. We found that the resulting YOLO model is accurate enough for vehicle detection during both daytime and nighttime.

After detection, the next step is to track detected vehicles in order to classify their trajectory as violating or not violating the law. We use a correlation tracking technique [61] that is more robust than other methods based on pixel or centroid tracking. With the use of the scale pyramid, the tracker can handle translation and scaling of objects in real time. Finally, after tracking, each vehicle track is analyzed to determine whether its trajectory passes through the junction while the traffic light is red.

Recognition of the traffic light status is performed by a separate thread named *Traffic-light Recognition Thread*. It processes video frames from the overview camera, simply thresholding the brightness and hue value of each lamp. Since the camera is fixed on a camera pole with a stationary view, the position of the traffic light appearing in the image can be pre-defined by the installer at setup time.

If a violation is found, *Main Thread* will capture several frames from the telephoto camera that focus on the lane the target vehicle is in. However, frames from the overview camera and the telephoto camera may not be synchronized in time, so there is a chance that the target vehicle may not be visible in the telephoto camera view at the right moment. From our observation, the time deviation (t_d) between overview and telephoto cameras can be modeled as approximately static. Hence, we created a dedicated separate process named *Frame Buffer* that buffers frames from the telephoto cameras for up to T milliseconds in order to guarantee that the target vehicle will be visible in at least one of the buffered frames. The parameters for the time deviation between cameras (t_d) and buffer length (T) are configured manually during system deployment.

4.2. Helmet Violation Detection

Similarly to the red-light violation detection system, the helmet violation detection system uses pure computer vision techniques to analyze video footage from IP cameras in real time. However, rather than processing video streams from multiple cameras, an instance of the helmet violation detection system only requires a stream capturing a single traffic lane from a single telephoto camera. Therefore, a deployment requires n cameras and n instances of the helmet violation detection system in accordance with the number of traffic lanes.

Figure 6b shows a flowchart of the helmet violation detection system. The first step is to detect and track motorcycles appearing in the telephoto camera. The motorcyclist was detected using a Haar cascade detector similar to that described in Section 4.1. We use a simple IOU (intersection over union) threshold to associate the target motorcycles in consecutive video frames. This method is adequate because the size of a motorcycle in a telephoto camera is proportionally large compared to the image size, and inter-vehicle occlusion rarely occurs when traffic is flowing. After a motorcycle is detected, the bounding box is horizontally split in half, then the top part is used for helmet/no-helmet classification, while the bottom half is used for license plate detection and recognition by the *Vehicle Information Extraction system* (described in Section 4.4). To analyze the top half of the image, we use AdaBoost cascade with Haar-like features in order to detect the head region inside the upper half of motorcyclist's bounding box. Then, the head region is classified as helmet or no-helmet using a support vector machine (SVM) classifier based on histogram of oriented gradients (HOG) features. Finally, a confidence score indicating the certainty of whether the tracked motorcyclist is not wearing helmet is calculated by the ratio of frames classified as no-helmet to the total number of frames in the track.

Our improvements to the helmet violation detection system using deep learning methods [62] can be described in two parts. First is the improvement of the motorcycle detector using YOLO, which similar to the vehicle detector for the red-light violation detection system previously described. Second, instead of detecting the head region and then classifying that region as a helmet or no-helmet, the improved module uses the entire top-half region of the motorcycle region for the classification. We use GoogLeNet as the classifier because it gives the best accuracy compared to other classifiers. The network is trained from scratch with two output classes: Motorcyclists and/or passengers with no helmet, and motorcyclists and/or passengers all wearing helmets.

4.3. Speed Violation Detection

The third analytic is vehicle speed violation detection. It is different from other analytics in that the system requires an extra sensor in addition to the IP cameras. We use directional radar speed guns or laser speed guns installed alongside the IP cameras in order to measure a vehicle's speed with a legally usable speed measurement device.

As shown in Figure 7a, the speed violation detection system has a similar workflow to that of the red-light violation detection system. The main difference is that the key criterion for a violation is an excess speed measurement from a speed gun rather than the traffic-light's status. However, there is another hidden process in the workflow after a violation is detected in particular, the identification of a putative over speeding vehicle in the video footage. We estimate the speed of each vehicle in the scene by considering the pixel displacement of vehicle and time difference between two consecutive frames. With geometric camera calibration relative to the ground plane, we can find the relationship between the pixel distance in the image and the actual distance on the road surface, allowing a vision-based measurement of vehicle speed. Finally, any vehicle speed measured as being over the limit by the speed guns are matched with the vehicle with the highest visual measurement speed. The assumption that these two measurement will match is valid since the radar speed guns used in this deployment always return the highest speed of multiple moving vehicles detected.

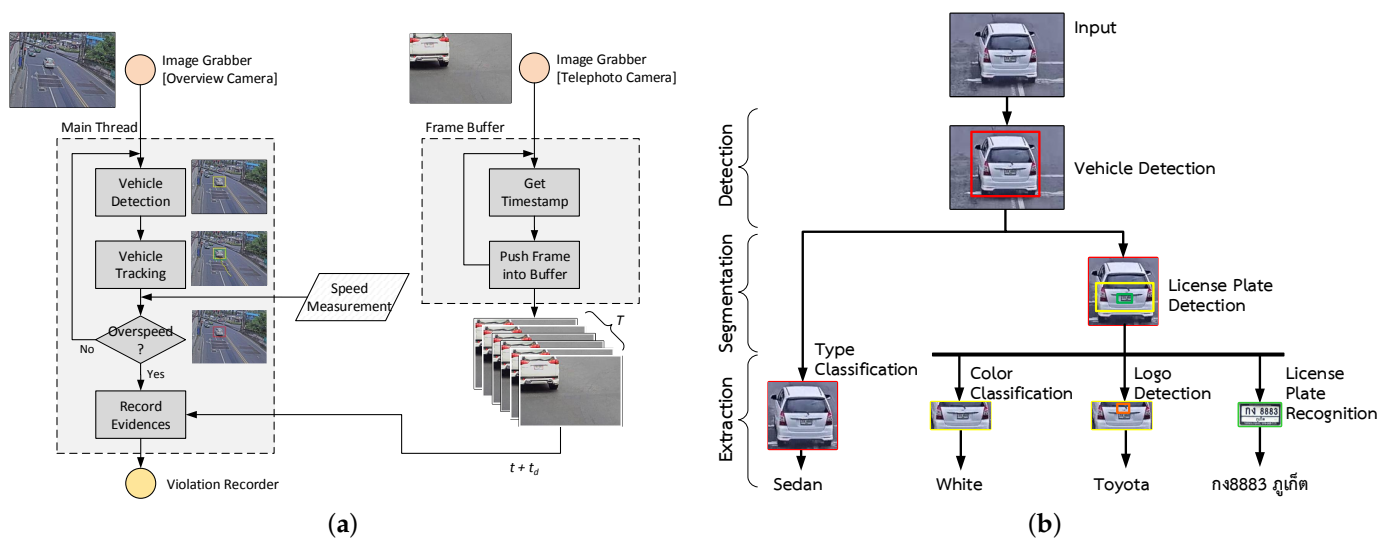


Figure 7. Flowcharts of (a) speed violation detection and (b) vehicle information extraction.

Similar to red-light and helmet analytics, the vehicle detection algorithm is upgraded to YOLO for the second phase of deployment of the speed violation detection system. Moreover, we upgraded the speed measurement sensor from a radar gun to a LIDAR gun for the second phase of the project in 2018. Radar is limited by wide antenna beam-widths, varying from 9° to 25° depending on the antenna size and the radar frequency. With the setup described in Section 3.1, the maximum distance from the radar gun to the target vehicle is about 40 m, which means that the beam cone can spread to 6–18 m, covering several traffic lanes. Hence, it is likely that there will usually be several vehicles in the beam. Even with the radar measurement to visual measurement matching algorithm described previously, it is not even proven with such a large field of view.

In the second version, we use a LIDAR with a much narrower beam (3–4 milliradians or 0.17 – 0.23°). However, this approach requires n LIDAR guns in order to cover an area with n traffic lanes. In this case, the logic flow is slightly changed from the first version. Instead of applying vehicle detection in the overview camera and identifying the vehicle with the highest visual speed measurement, the new speed violation detection system process the n telephoto camera streams directly. Each LIDAR speed gun is mapped one-to-one with a telephoto camera and each camera focuses on a single lane, so an over-speed vehicle can be simply matched with the fastest vehicle detected visually in the corresponding view.

4.4. Vehicle Information Extraction

As mentioned in Section 3.2.2, the vehicle information extraction system (*VInfo*) is a special analytic module dedicated for the automated extraction of data about violating vehicles. It extracts the vehicle type, color, brand, and license plate number based on evidence images provided by the other analytics.

There are three main steps in the vehicle information extraction workflow, as shown in Figure 7b: Detection, segmentation, and extraction. In the first step, the system finds the location of the target vehicle in the input evidence images. Similar to the detectors described in Section 4.1, the detection process in the first deployment phase was performed by a Haar-like AdaBoost cascade classifier, and in the second phase, it was performed using a YOLO detector.

In the segmentation process, *VInfo* crops several regions of a target vehicle's bounding box for further extraction processes. The first segment is the vehicle bounding box, which is used for vehicle type classification. The other two segments are the license plate region and the rear trunk region, which is extracted relative to the license plate bounding box. The license plate region is used for license number recognition, while the rear trunk region

is used for color and brand classification. As for the vehicle, we used the Haar cascade detector to find the license plate region in the first phase of deployment, which was changed to a YOLO detector in the second phase.

VInfo uses different techniques for the extraction of vehicle attributes from these three regions. The methods adopted in the first version are based on the AdaBoost cascade classifier plus classical computer vision techniques. In the second version, we switched to convolutional neural networks (CNNs). The implementation details of each extraction method are given as follows:

- *Type Classification*: In the first deployment phase, the whole vehicle region is passed to a HOG feature extractor then a SVM classifier trained to classify the vehicle into one of six types (sedan, pickup, van, SUV, bus, and truck), while motorcycles are detected with a separate Haar cascade model. In the second phase, we improved this classification by replacing the HOG feature extractor and SVM classifier with GoogLeNet;
- *Color Classification*: In the first deployment phase, we convert the rear trunk region from the RGB to the HSV color space, and then we use a SVM classifier to identify the vehicle color based on a HSV color histogram with $32 \times 16 \times 32$ bins. In the second phase, the classification method is changed to GoogLeNet, which takes the whole vehicle region as an input instead of the rear trunk region and classifies the vehicle image to eight output classes (white, black, gray, red, pink, green, blue, and yellow);
- *Brand Classification*: In the first deployment phase, vehicle brand classification is performed by locating and classifying the vehicle logo appearing on the vehicle's rear trunk region. The logo detection is based on a custom Haar cascade detector, and the logo classification is performed with a SVM classifier using HOG features. In the second phase, we trained YOLO to detect and classify 15 different vehicle logos appearing in the whole vehicle region;
- *License Plate Recognition*: In the first phase, the license plate region is passed through the binarization algorithm proposed by Wolf et al. [63]. The bounding box of each character is obtained from a connected component analysis and then each connected component is classified by a SVM classifier using HOG features. The improved OCR method in the second phase applies a single YOLO detector for both the detection and classification of the characters appearing in the license plate region. We also trained the network further to detect and classify the province name on Thai license plates. Since the province name characters are visibly small compared to the license number, it making it challenging for the system to recognize each single character, we trained YOLO to detect each entire province name as a whole instead of detecting individual characters and combining them to find a matching province name. Finally, our trained YOLO can detect 125 classes of symbols and words appearing on Thai license plates, including 38 Thai characters, 9 Arabic digits, and 78 province names.

5. Results

The first phase of deployment was launched in 2017 under funding from the Safer Roads Foundation (SRF), with the goal of urging drivers to pay attention to road safety and to decrease the number of accidents in the Phuket province. Using the system architecture described in Section 3 and implementing analytics as explained in Section 4, we deployed DRSSA at five intersections on Phuket island. The precise locations are given in Figure 8 (red map pins), and the latitude-longitude coordinates of each location are listed in Table 1. The distributed, modular structure of the sensors and processors in the edge and processor layer enabled us to shorten the total length of optical fiber required from 63.5 km (if using a centralized processing architecture with a high-bandwidth network linking every node) to 23.2 km. This enabled installation cost savings of around \$120,000 USD. This estimate is based on the price of optical fiber per kilometer (\sim \$3000/km) and the distance between nodes in the deployment.

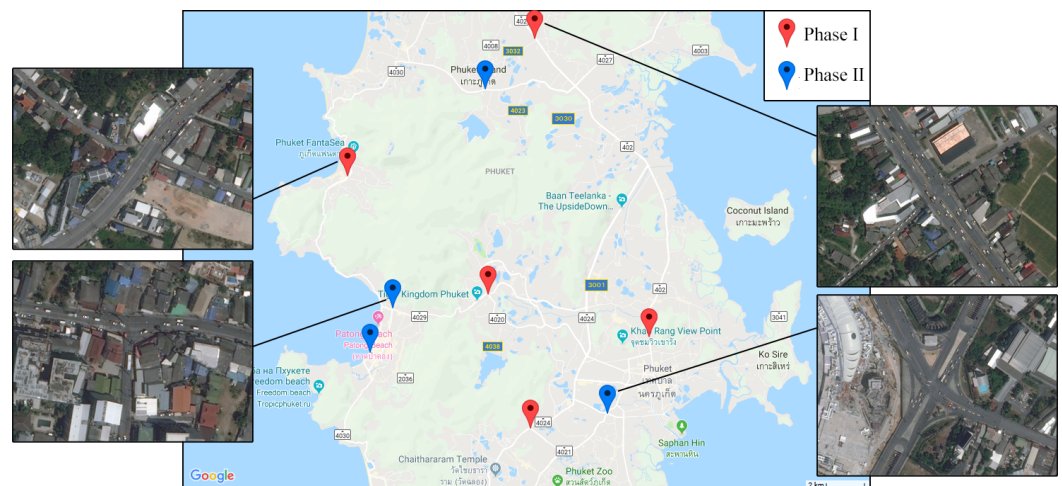


Figure 8. Locations of DRSSA nodes deployed in Phuket, Thailand.

Table 1. Location of traffic violation detection system deployments.

Code	Location	Lat/Long
Phase I		
LP	Li-Pond/Khao-Lan Intersection, Thalang	8°00′25.8″ N 98°20′33.2″ E
KL	4233/Hua-Kuan-Tai Intersection, Kathu	7°56′59.5″ N 98°17′06.8″ E
SK	See-Kor/Kah-Too Intersection, Kathu	7°54′38.6″ N 98°20′01.5″ E
KM	Komaraphat/Thepkasatree Junction, Mueang	7°53′46.9″ N 98°23′21.5″ E
KW	Chaofah/Kwang Junction, Mueang	7°51′54.2″ N 98°20′53.6″ E
Phase II		
MN	Ban Mah-nik School, Si Sunthon, Thalang	7°58′47.1″ N 98°19′59.8″ E
AS	Aomsin Bank, Patong, Kathu	7°54′14.2″ N 98°18′18.7″ E
PT	Taweewong Rd, Patong, Kathu	7°53′15.9″ N 98°17′30.1″ E
DR	Dawroong Wittaya School Intersection, Wichit, Mueang	7°52′11.5″ N 98°22′29.7″ E

With the layered structure of DRSSA, system maintenance and troubleshooting are also simplified. If a particular deployed analytic node is down, the system administrator can diagnose the problem by considering the status of the components, layer by layer. Furthermore, system repair and maintenance can be performed without downtime. Application servers in the application layer can be temporarily shut down for maintenance and upgrade while the analytic processors are still operating and storing violations in their local database. On the other hand, maintenance at a particular deployed location does not affect end users, since officers can still access violations and issue tickets for cases that have already been uploaded to the application server.

The second phase of deployment, in 2018, was part of a “Smart Phuket” project funded by Thailand’s National Electronics and Computer Technology Center (NECTEC) with the goal of establishing many smart city technologies in Phuket. DRSSA was chosen for expansion as part of the project’s public safety mission and to demonstrate the scalability of DRSSA. We improved each analytic as described in Section 4 and deployed the improved versions at four new locations, as drawn in Figure 8 with blue map pins.

A key feature of DRSSA is that different versions of the analytic modules can be interchanged or coexist in both the development and production environments. With a modular analytic structure and interfaces for connecting to components in other layers, as explained in Section 3.2, developers can modify the algorithm workflow for a particular analytic without affecting sensor connectivity or system integration. Moreover, in a specific deployment, processor nodes with newer analytic modules can be integrated with the

existing system without any change to other components. By allowing different versions of analytics to coexist on the same processing machine, depending on the resources available at a particular site, a DRSSA processor node can balance the number of instances between the low-resource earlier version and the high-performance but resource-demanding later version. This helps simplify system sizing and enables the upgradability and scalability of the system.

Over the two years that DRSSA and our video analytic modules were deployed, we also faced issues regarding hardware failures such as broken cameras and network links. In most cases, only the sub-systems that rely on the broken hardware were affected while the overall system continued to operate. However, replacing inoperable equipment requires new budget allocation, making the project unsustainable. To solve this issue, from the beginning of the project in 2017, we put in place structuring for cooperation between the municipal government and the corresponding municipal police departments, signing an agreement to jointly share and utilize the revenue from traffic tickets generated by the system. These agreements stipulate that the income from traffic fines should be split equally between the two departments and used to cover expenses for system operation and maintenance. Local police stations utilize this new revenue stream for materials needed in the ticket issuing process, such as printer cartridges and postal fees, while the municipal offices use this new revenue stream for hardware replacement and maintenance. Over three years, this arrangement has moved slowly due to rigid government processes and constant change of representatives to the municipal councils, but now only a few municipalities have yet to sign the cooperation agreement.

Apart from the architectural advantages of DRSSA, the adoption of CNNs in the second phase of deployment resulted in significant improvement in every analytic's performance. We built an evaluation dataset by taking sample footage from all deployed locations. The samples were selected to cover lighting conditions over the entire day and to contain every traffic condition such as free-flow and rush hour traffic. As shown in Table 2, the precision and recall of violation detection analytics are substantially increased, and the precision of the information extraction analytic is also improved. The first-phase analytics suffer from uncertainty in dynamic system environments such as low light conditions at night and occlusion of target vehicles due to traffic congestion. These factors cause the phase 1 DRSSA analytics to have low recall.

Table 2. Accuracy of video analytics in DRSSA.

Analytics	Classical Image Processing			Convolutional Neural Network		
	Precision	Recall	F_1	Precision	Recall	F_1
Red-light	0.85	0.54	0.66	0.85	0.82	0.83
Helmet	0.84	0.45	0.58	0.93	0.80	0.86
Speed	0.75	0.31	0.43	0.91	0.81	0.86
Vehicle Information Extraction						
License Number	0.85			0.88		
Type	0.77			0.89		
Color	0.75			0.82		
Brand	0.84			0.90		

For analytics that do not utilize image processing, speed violation detection is affected by limitations in the sensor, since it is hard to pinpoint target vehicles using a wide-beam radar installed on a high pole. Moreover, there is usually more than one vehicle appearing in evidence images, which might lead to challenges by the ticket receiver. Hence, in the first deployment, the speed violation detection analytic was not used for issuing traffic tickets, but was used for data collection and statistical analysis instead.

In the second phase, the accuracy of every violation detection analytics improved due to adopting YOLO for vehicle detection and GoogLeNet for classification. The effects

of uncertainties in the environment on the analytics' accuracy can be relatively easily eliminated by using a dataset with various lighting and traffic conditions gathered in the first phase as a training set for the neural network model. The results indicate a significant improvement in both precision and recall for all analytics and the accuracy of vehicle information extraction.

6. Impact on Driver Compliance and Road Fatalities

In order to maximize impact on road safety, at the beginning of the project, we targeted hazardous road locations reported to have a high number of traffic accidents and fatalities every year. Most of the sites are located where highways cut through high population density areas, as shown in Figure 8. To analyze the impact of DRSSA on road users' compliance with traffic laws, we analyze the number of violations detected by DRSSA as a function of time. Furthermore, we examine the impact of DRSSA on road safety in Phuket by comparing the road accident statistics before and after system deployment.

The first phase started in mid-2017, and took six months to establish cooperation with government offices and obtain required permits necessary for the police to issue tickets based on the system. However, this delay period provided an opportunity to monitor the impact of traffic cameras on driver behavior in terms of the number of violations detected over time before the ticketing system came online as camera poles and warning signs were already installed and visible to road users. We did not observe any significant change in the number of violations (as represented by the red shaded area in Figure 9) from the end of 2017 to the beginning of 2018. The first actual traffic citation was issued in February 2018. The number of citations issued each month is represented by the gray shaded area in Figure 9. The red dotted lines and gray dashed lines are trend lines showing a basic 3-month moving average of the number of violations and the number of citations respectively. In case of red-light and helmet violations, the number of violations in the months after the police started issuing traffic tickets decreased gradually: The number of red-light violations dropped from ~8000 cases/month in the beginning of 2018 to ~6000 cases/month in the end of 2019 and the number of helmet violations dropped from ~100,000 cases/month to ~90,000 cases/month in the same period of time. However, we can observe that this reduction did not occur for speed violations. This is due to the fact that there were no speed citations issued, due to the concern described in Section 5. This can imply that only the installation of traffic cameras without real enforcement (actual tickets) did not create an impact on local road users' awareness and behavior.

We also consider statistical data from other sources in order to measure the influence of the system on the behavior of local road users. According to the ThaiRoads Foundation, an organization established to analyze road traffic accident data and improve road safety policies in Thailand, the percentage of motorcycle drivers and passengers wearing helmets in Phuket (measured by visual survey) has been steady at 60% since 2017 (as shown by blue line with square shape markers in Figure 10). In urban and suburban areas, where DRSSA is deployed, compliance is at 68% and 61%, respectively, compliance in rural areas not covered by DRSSA dropped from 50% in 2017 to 48% in 2019. Despite the significant drops in the number of helmet violations detected by DRSSA, the overall percentage of motorcycle drivers wearing helmets over the entire province has not changed. The reason is that compared to the total number of motorcycle drivers not wearing helmets, the number of deployments and tickets generated by the system has been too low to affect compliance province-wide. We observed a significant impact at specific sites only where DRSSA is deployed.

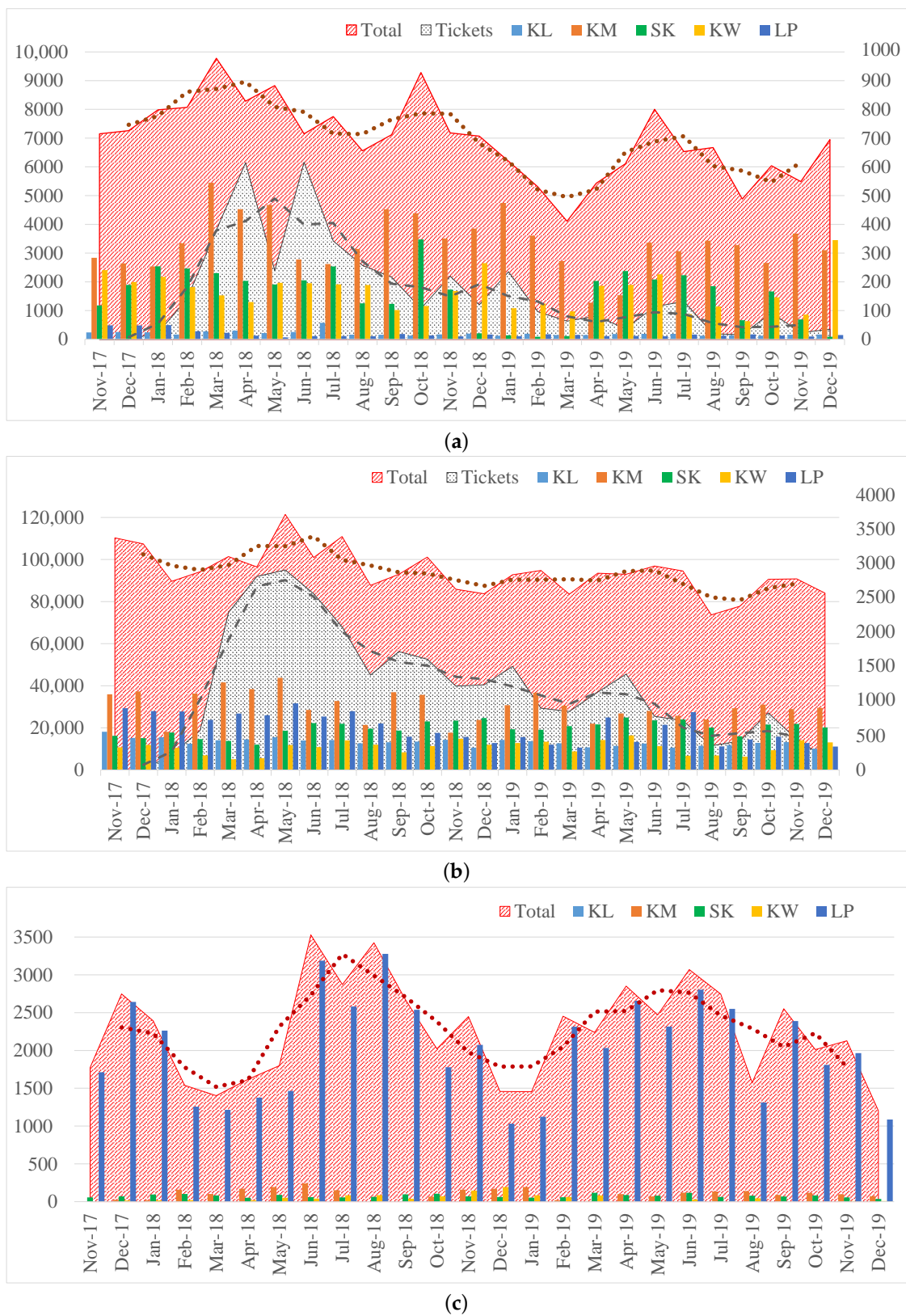


Figure 9. Number of monthly (a) red-light violations, (b) helmet violations, and (c) speed violations from November 2017 to August 2019.

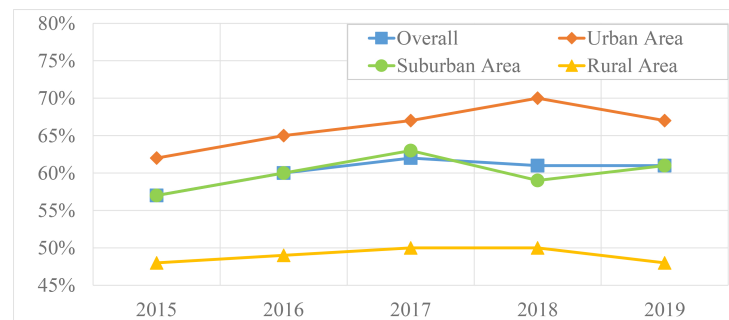
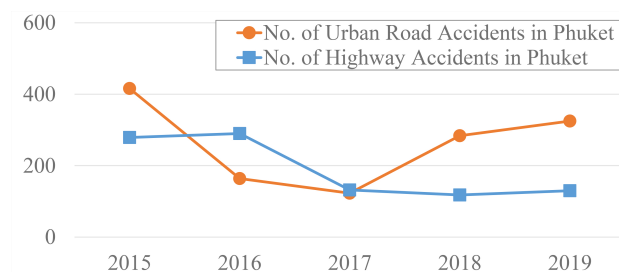
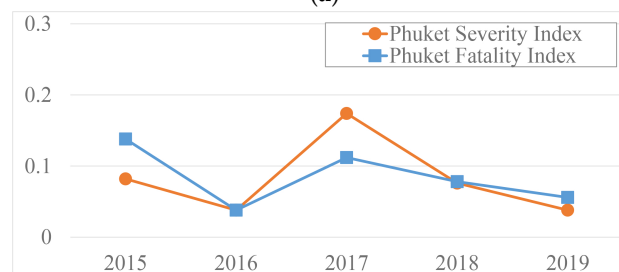


Figure 10. Percentage of motorcyclists and passengers wearing helmets in Phuket from 2015 to 2019.

Lastly, to understand the impact of the deployment on road safety in Phuket, we also analyze the number of annual road traffic accidents and fatalities compiled by the ThaiRoads Foundation for the province. Figure 11a shows that, since 2017, the number of highway accidents in Phuket has decreased, while the number of urban accidents has risen. DRSSA was deployed primarily on highways, so it likely had impact on highway accidents but not traffic accidents in the denser urban areas. However, clearly, highway accidents are more severe than urban accidents, the reason for the choice of highway locations for the deployment in the first place. Furthermore, as shown in Figure 11b, road traffic accident severity dropped by a wide margin in 2018. The severity index is defined as the number of persons killed per accident and the fatality index is the number of deaths in proportion to the total number of deaths and injuries. We conclude that DRSSA, through a modest investment in infrastructure and government process changes, has had a positive impact on Phuket road safety. The system encourages road users to comply with traffic laws at the site where it is deployed, corroborated by the gradually decreasing number of violations detected by our video analytic system and the lower number of highway accidents after system deployment.



(a)



(b)

Figure 11. Accident statistics in Phuket. (a) Annual road accidents. (b) Severity and fatality rates for accidents in Phuket from 2015 to 2019.

7. Conclusions

We proposed the Distributed Road Safety System Architecture (DRSSA), which is designed as a cost-efficient traffic violation enforcement system. By decomposing the system components into layers, DRSSA decouples analytic processors from application servers and eliminates the need for an expensive high-bandwidth network linking them. Moreover,

the modular software architecture of DRSSA allows flexibility, scalability, and resource optimization for the analytics deployed to the system. We also presented the implementations of video analytics for three types of traffic violations: Red-light running, motorcycling without a helmet, and over-speeding. Two versions of the video analytics were implemented and deployed in Phuket, Thailand. Results indicate that the second version of video analytics using CNNs—YOLO as a detector and GoogLeNet for the classifiers—could handle visual uncertainties and yielded better accuracy than the first version, which relies on less robust classical image processing methods. We studied the impact of automated traffic violation detection on driver compliance and road safety in Phuket, Thailand. The data show that the combination of automated systems and in-person enforcement by authorities clearly affected road users' behavior at deployment areas. However, to affect broad changes across the whole city area, more intense enforcement would be required.

Finally, with cooperation and support from multiple government agencies, we demonstrated the implementation of a sustainable system that strengthens traffic law enforcement in Phuket. In future, this system can be scaled up with a modest investment in order to maximize the level of compliance with traffic laws and to improve road safety across the province.

Author Contributions: The authors confirm contribution to the paper as follows: Writing—original draft: S.L.; Writing—review & editing: M.N.D.; Methodology: S.L., M.N.D., R.M., V.T., A.S.; Software: S.L., R.M., V.T., A.C., A.S.; Supervision: M.N.D., W.S., M.E. All authors have read and agreed to the published version of the manuscript.

Funding: This work was initiated within a project funded by the Safer Roads Foundation (SRF) to improve road safety in Phuket, Thailand and extended within the “Smart Phuket” project funded by Thailand’s National Electronics and Computer Technology Center (NECTEC).

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from the ThaiRoads Foundation and are available at <http://trso.thairoads.org/statistic> with the permission of the ThaiRoads Foundation.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

APM	Analytic Process Manager
CA	Certificate Authority
CNNs	Convolutional Neural Networks
DRSSA	Distributed Road Safety System Architecture
FPS	Frame per Second
GMM	Gaussian mixture model
HOG	Histogram of Oriented Gradients
mAP	Mean Average Precision
SIFT	scale-invariant feature transform
SURF	speeded-up robust feature
SVM	Support Vector Machine
VANETs	Vehicular Ad-hoc Networks
VInfo	Vehicle Information Extraction

References

1. *Global Status Report on Road Safety 2018*; World Health Organization: Geneva, Switzerland, 2018.
2. Tay, R. Speed Cameras Improving Safety or Raising Revenue? *JTEP* **2010**, *44*, 247–257.
3. De Pauw, E.; Daniels, S.; Brijs, T.; Hermans, E.; Wets, G. An evaluation of the traffic safety effect of fixed speed cameras. *Saf. Sci.* **2014**, *62*, 168–174. [[CrossRef](#)]
4. Caldeira, J.; Fout, A.; Kesari, A.; Sefala, R.; Walsh, J.; Dupre, K.; Khaefi, M.R.; Hodge, G.; Pramestri, Z.A.; Imtiyazi, M.A. Improving Traffic Safety Through Video Analysis in Jakarta, Indonesia. In *Proceedings of the SAI Intelligent Systems Conference, London, UK, 5–6 September 2019*; pp. 642–649.

5. Cheung, S.Y.; Coleri, S.; Dunder, B.; Ganesh, S.; Tan, C.W.; Varaiya, P. Traffic measurement and vehicle classification with single magnetic sensor. *Transp. Res. Rec.* **2005**, *1917*, 173–181. [\[CrossRef\]](#)
6. Cheeverunothai, P.; Wang, Y.; Nihan, N.L. Identification and correction of dual-loop sensitivity problems. *Transp. Res. Rec.* **2006**, *1945*, 73–81. [\[CrossRef\]](#)
7. Li, H.; Dong, H.; Jia, L.; Ren, M. Vehicle classification with single multi-functional magnetic sensor and optimal MNS-based CART. *Measurement* **2014**, *55*, 142–152. [\[CrossRef\]](#)
8. Meta, S.; Cinsdikici, M.G. Vehicle-classification algorithm based on component analysis for single-loop inductive detector. *IEEE Trans. Veh. Technol.* **2010**, *59*, 2795–2805. [\[CrossRef\]](#)
9. Tok, A.; Ritchie, S.G. Vector Classification of Commercial Vehicles Using a High Fidelity Inductive Loop Detection System. In Proceedings of the 89th Annual Meeting Transportation Research Board, Washington, DC, USA, 10–14 January 2010; pp. 1–22.
10. Fischer, J.; Menon, A.; Gorjestani, A.; Shankwitz, C.; Donath, M. Range sensor evaluation for use in cooperative intersection collision avoidance systems. In Proceedings of the 2009 IEEE Vehicular Networking Conference (VNC), Tokyo, Japan, 28–30 October 2009; pp. 1–8.
11. Raja Abdullah, R.S.A.; Abdul Aziz, N.H.; Abdul Rashid, N.E.; Ahmad Salah, A.; Hashim, F. Analysis on target detection and classification in LTE based passive forward scattering radar. *Sensors* **2016**, *16*, 1607. [\[CrossRef\]](#)
12. Lee, H.; Coifman, B. Using LIDAR to validate the performance of vehicle classification stations. *J. Intell. Transp. Syst.* **2015**, *19*, 355–369. [\[CrossRef\]](#)
13. Lee, H.; Coifman, B. Side-fire lidar-based vehicle classification. *Transp. Res. Rec.* **2012**, *2308*, 173–183. [\[CrossRef\]](#)
14. Asborn, M.I.; Burris, C.G.; Hernandez, S. Truck body-type classification using single-beam LiDAR sensors. *Transp. Res. Rec.* **2019**, *2673*, 26–40. [\[CrossRef\]](#)
15. Chen, Z.; Ellis, T.; Velastin, S.A. Vehicle detection, tracking and classification in urban traffic. In Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, AK, USA, 16–19 September 2012; pp. 951–956.
16. Mithun, N.C.; Rashid, N.U.; Rahman, S.M. Detection and classification of vehicles from video using multiple time-spatial images. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1215–1225. [\[CrossRef\]](#)
17. Unzueta, L.; Nieto, M.; Cortés, A.; Barandiaran, J.; Otaegui, O.; Sánchez, P. Adaptive multicue background subtraction for robust vehicle counting and classification. *IEEE Trans. Intell. Transp. Syst.* **2011**, *13*, 527–540. [\[CrossRef\]](#)
18. Dong, Z.; Wu, Y.; Pei, M.; Jia, Y. Vehicle type classification using a semisupervised convolutional neural network. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2247–2256. [\[CrossRef\]](#)
19. Karaimer, H.C.; Cinaroglu, I.; Bastanlar, Y. Combining shape-based and gradient-based classifiers for vehicle classification. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain, 15–18 September 2015; pp. 800–805.
20. Huttunen, H.; Yancheshmeh, F.S.; Chen, K. Car type recognition with deep neural networks. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 19–22 June 2016; pp. 1115–1120.
21. Adu-Gyamfi, Y.O.; Asare, S.K.; Sharma, A.; Titus, T. Automated vehicle recognition with deep convolutional neural networks. *Transp. Res. Rec.* **2017**, *2645*, 113–122. [\[CrossRef\]](#)
22. Zhao, D.; Chen, Y.; Lv, L. Deep reinforcement learning with visual attention for vehicle classification. *IEEE Trans. Cogn. Dev. Syst.* **2016**, *9*, 356–367. [\[CrossRef\]](#)
23. Theagarajan, R.; Pala, F.; Bhanu, B. EDnN: Ensemble of deep networks for vehicle classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 33–40.
24. Kim, P.K.; Lim, K.T. Vehicle type classification using bagging and convolutional neural network on multi view surveillance image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 41–46.
25. Liu, W.; Zhang, M.; Luo, Z.; Cai, Y. An ensemble deep learning method for vehicle type classification on visual traffic surveillance sensors. *IEEE Access* **2017**, *5*, 24417–24425. [\[CrossRef\]](#)
26. Chang, J.; Wang, L.; Meng, G.; Xiang, S.; Pan, C. Vision-based occlusion handling and vehicle classification for traffic surveillance systems. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 80–92. [\[CrossRef\]](#)
27. ElHakim, R.; Abdelwahab, M.; Eldesokey, A.; ElHelw, M. Traffisense: A smart integrated visual sensing system for traffic monitoring. In Proceedings of the 2015 SAI Intelligent Systems Conference (IntelliSys), London, UK, 10–11 November 2015; pp. 418–426.
28. Billones, R.K.C.; Bandala, A.A.; Sybingco, E.; Lim, L.A.G.; Dadios, E.P. Intelligent system architecture for a vision-based contactless apprehension of traffic violations. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON), Singapore, 22–25 November 2016; pp. 1871–1874.
29. Chen, A.; Khorashadi, B.; Chuah, C.N.; Ghosal, D.; Zhang, M. Smoothing vehicular traffic flow using vehicular-based ad hoc networking & computing grid (VGrid). In Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference, Toronto, ON, Canada, 17–20 September 2006; pp. 349–354.
30. Ahmed, S.H.; Yaqub, M.A.; Bouk, S.H.; Kim, D. Towards content-centric traffic ticketing in VANETs: An application perspective. In Proceedings of the 2015 Seventh International Conference on Ubiquitous and Future Networks, Sapporo, Japan, 7–10 July 2015; pp. 237–239.

31. Mallisery, S.; Pai, M.M.; Ajam, N.; Pai, R.M.; Mouzna, J. Transport and traffic rule violation monitoring service in ITS: A secured VANET cloud application. In Proceedings of the 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2015; pp. 213–218.
32. Özkul, M.; Çapuni, I. Police-less multi-party traffic violation detection and reporting system with privacy preservation. *IET Intell. Transp. Syst.* **2018**, *12*, 351–358. [[CrossRef](#)]
33. Bouchelaghem, S.; Omar, M. Reliable and secure distributed smart road pricing system for smart cities. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 1592–1603. [[CrossRef](#)]
34. Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S.; Koscher, K.; Czeskis, A.; Roesner, F.; Kohno, T. Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011; pp. 447–462.
35. Kelarestaghi, K.B.; Foruhandeh, M.; Heaslip, K.; Gerdes, R. Survey on vehicular ad hoc networks and its access technologies security vulnerabilities and countermeasures. *arXiv* **2019**, arXiv:1903.01541.
36. Hussain, R.; Lee, J.; Zeadally, S. Trust in VANET: A Survey of Current Solutions and Future Research Opportunities. *IEEE Trans. Intell. Transp. Syst.* **2020**. [[CrossRef](#)]
37. *The National Intersection Safety Problem*; Brief Issues #2; FHWA-SA-10-005; U. S. Department of Transportation, Federal Highway Administration: Washington, DC, USA, 2009.
38. Loce, R.P.; Bala, R.; Trivedi, M. A computer vision framework for transportation applications. In *Computer Vision and Imaging in Intelligent Transportation Systems*; Wiley: Hoboken, NJ, USA, 2017; pp. 7–12.
39. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–26 June 2005; pp. 886–893.
40. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition (CVPR), Kauai, HI, USA, 8–14 December 2001; pp. 511–518.
41. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *IJCV* **2004**, *60*, 91–110. [[CrossRef](#)]
42. Mandellos, N.A.; Keramitsoglou, I.; Kiranoudis, C.T. A background subtraction algorithm for detecting and tracking vehicles. *Expert Syst. Appl.* **2011**, *38*, 1619–1631. [[CrossRef](#)]
43. Su, X.; Khoshgoftaar, T.M.; Zhu, X.; Folleco, A. Rule-based multiple object tracking for traffic surveillance using collaborative background extraction. In Proceedings of the ISVC, Lake Tahoe, NV, USA, 26–28 November 2007; pp. 469–478.
44. Li, Y.; Li, Z.; Tian, H.; Wang, Y. Vehicle detecting and shadow removing based on edged mixture Gaussian model. *IFAC* **2011**, *44*, 800–805. [[CrossRef](#)]
45. Gao, T.; Liu, Z.G.; Gao, W.C.; Zhang, J. Moving vehicle tracking based on SIFT active particle choosing. In Proceedings of the International Conference on Neural Information Processing (ICONIP), Auckland, New Zealand, 25–28 November 2008; pp. 695–702.
46. Hsieh, J.W.; Chen, L.C.; Chen, D.Y. Symmetrical SURF and its applications to vehicle detection and vehicle make and model recognition. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 6–20. [[CrossRef](#)]
47. Yuan, Q.; Thangali, A.; Ablavsky, V.; Sclaroff, S. Learning a family of detectors via multiplicative kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *33*, 514–530. [[CrossRef](#)] [[PubMed](#)]
48. Chang, W.C.; Cho, C.W. Online boosting for vehicle detection. *IEEE Trans. Syst. Man Cybern. Syst. Hum. Part (Cybern.)* **2009**, *40*, 892–902. [[CrossRef](#)]
49. Gleason, J.; Nefian, A.V.; Bouyssounousse, X.; Fong, T.; Bebis, G. Vehicle detection from aerial imagery. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 2065–2070.
50. Elkerdawi, S.M.; Sayed, R.; ElHelw, M. Real-time vehicle detection and tracking using Haar-like features and compressive tracking. In Proceedings of the ROBOT2013: First Iberian Robotics Conference, Madrid, Spain, 28–29 November 2013; pp. 381–390.
51. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [[CrossRef](#)]
52. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
53. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
54. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
55. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
56. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
57. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014; pp. 580–587.
58. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.

59. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [[CrossRef](#)] [[PubMed](#)]
60. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
61. Danelljan, M.; Häger, G.; Khan, F.; Felsberg, M. Accurate scale estimation for robust visual tracking. In Proceedings of the British Machine Vision Conference, Nottingham, UK, 1–5 September 2014.
62. Chairat, A.; Dailey, M.; Limsoonthrakul, S.; Ekpanyapong, M.; Dharma Raj, K.C. Low Cost, High Performance Automatic Motorcycle Helmet Violation Detection. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Snowmass Village, CO, USA, 1–5 March 2020; pp. 3560–3568.
63. Wolf, C.; Jolion, J.M.; Chassaing, F. Text localization, enhancement and binarization in multimedia documents. In Proceedings of the Object Recognition Supported by User Interaction for Service Robots, Quebec City, QC, Canada, 11–15 August 2002; pp. 1037–1040.