

Tunnel Traffic Enforcement Using Visual Computing and Field-Programmable Gate Array-Based Vehicle Detection and Tracking [†]

Yi-Chen Lin ¹ and Rey-Sern Lin ^{2,*}

¹ Department of Electrical Engineering, National Taiwan Normal University, Taipei 106, Taiwan; 40940112s@std.ntnu.edu.tw

² Department of Public Administration, Chung Hua University, Hsinchu 30012, Taiwan

* Correspondence: rslin@g.chu.edu.tw

[†] Presented at the 2024 IEEE 6th Eurasia Conference on IoT, Communication and Engineering, Yunlin, Taiwan, 15–17 November 2024.

Abstract: Tunnels are commonly found in small and enclosed environments on highways, roads, or city streets. They are constructed to pass through mountains or beneath crowded urban areas. To prevent accidents in these confined environments, lane changes, slow driving, or speeding are prohibited on single- or multi-lane one-way roads. We developed a foreground detection algorithm based on the K-nearest neighbor (KNN) and Gaussian mixture model and 400 collected images. The KNN was used to gather the first 200 image data, which were processed to remove differences and estimate a high-quality background. Once the background was obtained, new images were extracted without the background image to extract the vehicle's foreground. The background image was processed using Canny edge detection and the Hough transform to calculate road lines. At the same time, the oriented FAST and rotated BRIEF (ORB) algorithm was employed to track vehicles in the foreground image and determine positions and lane deviations. This method enables the calculation of traffic flow and abnormal movements. We accelerated image processing using xfOpenCV on the PYNQ-Z2 and FPGA Xilinx platforms. The developed algorithm does not require pre-labeled training models and can be used during the daytime to automatically collect the required footage. For real-time monitoring, the proposed algorithm increases the computation speed ten times compared with YOLO-v2-tiny. Additionally, it uses less than 1% of YOLO's storage space. The proposed algorithm operates stably on the PYNQ-Z2 platform with existing surveillance cameras, without additional hardware setup. These advantages make the system more appropriate for smart traffic management than the existing framework.



Academic Editors: Teen-Hang Meen, Chi-Ting Ho and Cheng-Fu Yang

Published: 25 April 2025

Citation: Lin, Y.-C.; Lin, R.-S. Tunnel Traffic Enforcement Using Visual Computing and Field-Programmable Gate Array-Based Vehicle Detection and Tracking. *Eng. Proc.* **2025**, *92*, 30. <https://doi.org/10.3390/engproc2025092030>

Copyright: © 2025 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/>).

Keywords: Gaussian mixture model; foreground detection; FPGA; traffic enforcement; smart traffic management

1. Introduction

Traffic enforcement technology utilizes advanced devices, such as cameras, automated detection equipment, and wireless data transmission, to assist law enforcement officers in managing traffic and reducing violations. This enhances enforcement efficiency and increases public safety [1]. Automated monitoring systems, such as electronic traffic lights and speed cameras, operate around the clock, accurately capturing violations and improving the efficiency of penalty enforcement [2,3].

Taiwan's high population density and limited space often result in traffic congestion. Therefore, effective enforcement is required as a deterrent for violators to improve traffic flow and reduce accidents. However, traditional methods such as spot checks are time-consuming and labor-intensive. Moreover, when faced with unpredictable factors such as weather, time, and location, manual enforcement becomes difficult to execute effectively. Tunnels are commonly found in Taiwan's highways, main roads, or in small, enclosed spaces within urban streets. They are typically constructed to pass through mountains or crowded urban areas. To prevent accidents in these confined environments, single or multi-lane one-way traffic lanes are strictly regulated, prohibiting arbitrary lane changes, slow driving, or speeding. Therefore, technology-based traffic violation detection within tunnels is essential.

To address issues in technology-based law enforcement, vehicle detection, road marking, and lane of road direction must be developed. A method for vehicle detection using deep learning techniques in tunnels was developed to solve the interference problems caused by poor lighting and complex backgrounds that affect traditional detection methods [4]. Special conditions within tunnels, such as shadows and light reflections, impact the accuracy of vehicle detection. Another deep learning-based vehicle detection system that utilizes convolutional neural networks (CNN) to extract features from tunnel images for the precise identification of vehicle positions and shapes [5]. The system significantly improves detection efficiency under low-light conditions for automated traffic management and accident prevention in tunnels. Recently, Ghanem et al. [6] solved the problem of lane detection under artificial lighting in tunnels and highways. Lane detection in such environments is affected by light reflections, shadows, and other factors, increasing the risk for drivers. They proposed an Internet of Things (IoT)-based framework that combines multiple sensors and deep learning techniques for lane detection. This approach enables accurate lane detection and enhances the reliability of autonomous driving and driver assistance systems.

Previous studies have shown that deep learning is crucial in vehicle and lane detection. Secondly, Sultana et al. [7] proposed a vision-based lane detection and tracking method. Even in complex environments with lighting variations, shadows, and weather conditions, image processing techniques and deep learning models analyze road images and enhance detection accuracy through multi-layer feature extraction techniques [8]. These results support the advancement of autonomous driving. To improve driving safety, Liu et al. [9] investigated the rear vehicle detection and tracking method in lane-changing assistance systems. They used cameras to capture images of rear vehicles and applied image processing techniques to identify vehicle positions and dynamics. The Kalman filter was used for precise vehicle tracking. Their method helps drivers understand the status of rear vehicles in real time when changing lanes, thereby reducing the risk of traffic accidents. Nguyen et al. [10] utilized image processing and deep learning techniques for real-time lane and surrounding vehicle position recognition. Their image processing method combines the Hough transform [11] for lane detection and vehicle recognition techniques based on color and shape, significantly improving the system's accuracy and response speed on highways and providing a safer lane-changing assistance system.

In Taiwan, research on tunnel technology enforcement has been conducted [8]. A visual system for vehicle detection and event monitoring was proposed for the Hsuehshan Tunnel to address the challenge of timely detection of traffic incidents. By using cameras, vehicle dynamics inside the tunnel can be captured. By employing image processing techniques, vehicle speed, position, and potential traffic accidents are detected. Lane detection using image processing techniques was effective.

These studies use advanced technologies to tackle vehicle and lane detection issues in various environments. Whether through deep learning or traditional image processing techniques, traffic safety and system stability are improved. By integrating the results of the aforementioned research, we designed a cost-effective, real-time monitoring, lightweight, versatile, and innovative traffic state monitoring system. By leveraging the performance advantages of the field-programmable gate array (FPGA) and the user-friendly design features of the PYNQ-Z2 platform, we reduced the weight of our intelligent traffic monitoring system and system implementation costs. With the advent of artificial intelligence (AI) technology, we combined roadside equipment for image capture and real-time computing to monitor real-time traffic flows and develop smart traffic systems. The lightweight algorithm, run on the FPGA platform, enables rapid synchronization of information for effective management, realizing AI-based applications such as mobile traffic event reporting, risk prediction, blind spot alerts, and early event warnings.

2. Background Knowledge

2.1. Foreground Detection

Foreground detection is the primary step in object tracking. The Gaussian mixture model (GMM) [4] is a classic algorithm for background construction. Since its introduction, improvements and applications have been made. In the GMM, the pixel belonging to the foreground or background is determined by (1):

$$R = \frac{p(BG|\vec{x}(t))}{p(FG|\vec{x}(t))} = \frac{p(\vec{x}(t)|BG)p(BG)}{p(\vec{x}(t)|FG)p(FG)} \quad (1)$$

where $\vec{x}(t)$ represents the pixel value in the image space at time t .

We assume that the probability of a certain point belonging to the background and foreground is the same, and the foreground object has a uniform distribution, i.e., $p(\vec{x}(t)|FG) = c_{FG}$. Therefore, to satisfy (2), the pixel belongs to the background.

$$p(\vec{x}(t)|BG) > c_{thr} = R_{c_{FG}} \quad (2)$$

where c_{thr} represents the threshold value, and $p(\vec{x}(t)|BG)$ is referred to as the background model.

The background model is estimated by training the set χ , represented as $\hat{p}(\vec{x}(t)|\chi, BG)$, which is dependent on the distribution of the training set. By selecting an appropriate time window T for the sample set at time t , $\chi_T = \{\vec{x}(t), \vec{x}(t+1), \dots, \vec{x}(t+T)\}$, the background model is updated and re-estimated with each new sample, χ_T and $\hat{p}(\vec{x}(t)|\chi, BG)$. A GMM with M adjustable distributions is used as (3):

$$\hat{p}(\vec{x}(t)|\chi, BG + FG) = \sum_{m=1}^M \hat{\pi}_m N(\vec{x}(t); \hat{\mu}_m, \hat{\sigma}_m^2 I) \quad (3)$$

where $\hat{\mu}_1, \dots, \hat{\mu}_M$ are the estimated Gaussian means, $\hat{\delta}_1, \dots, \hat{\delta}_M$ are the estimated Gaussian standard deviations, and $\hat{\pi}_1, \dots, \hat{\pi}_M$ are non-negative weights that sum up to 1.

2.2. Object Detection

Object detection is used to determine whether the target objects appear in the image. Generally, bounding boxes are used to label each object separately. The method is a fast algorithm for labeling object contours based on binary images [12]. The process involves

tracking only the outermost boundaries of objects and describing them using a topological approach. This method is advantageous due to its rapid computation, making it highly suitable for real-time object detection.

2.3. Object Tracking

Object tracking is important in visual surveillance. Most research is based on the oriented fast and rotated BRIEF (ORB). ORB features offer advantages such as efficient computation, rotation invariance, and noise resistance. The object tracking method is based on ORB features, which compute the conditions for the closest spatial distance in consecutive adjacent frames [13]. The algorithm is as follows.

Step 1: Obtain frame i and the next frame $(i + 1)$.

Step 2: Predict the next search window $S_{i+1}(u_{i+1}, v_{i+1}, w', h')$ based on the current tracked object location box $Rec_i(x_i, y_i, w, h)$, where $(u_{i+1}, v_{i+1}) = (x_i, y_i) + \vec{M}_i$, $w' = w + \alpha$, $h' = h + \alpha$, and $\vec{M}_i = (x_i, y_i) - (x_{i-1}, y_{i-1})$.

Step 3: Detect ORB key points in Rec_i and S_{i+1} , and extract the corresponding binary descriptors D_i and D_{i+1} .

Step 4: Match between D_i and D_{i+1} , then get two matched subsets of key points MK_i and MK_{i+1} .

Step 5: Calculate the geometric centers of MK_i and MK_{i+1} , denoted as C_i and C_{i+1} .

Step 6: Update the location based on the motion vector. If no match is found between D_i and D_{i+1} , use the last motion vector $\vec{M}_i$.

Step 7: $i = i + 1$ and repeat *Step 1*.

2.4. FPGA

FPGA is a semiconductor device that is reconfigured by the user to program circuit functionality directly on the hardware. The core characteristic of FPGA is its parallel processing capability, allowing multiple logic units to perform different operations simultaneously. This feature is particularly advantageous in applications that require high performance and low latency. Therefore, FPGAs have significant benefits in the real-time processing of large amounts of data, such as signal processing [14], communication, image processing, and cryptography. In communication, FPGAs are used to handle high-speed data streams, such as digital signal processing in wireless communication base stations, 5G networks, and data converters in optical communications. In autonomous driving and AI, FPGAs can also accelerate the inference process of deep learning models, providing low-latency processing capabilities in scenarios with strict timing requirements. Additionally, FPGAs are applied in aerospace, medical electronics, industrial automation, and high-frequency trading systems in the financial sector, where there are high demands for reliability and performance. The advantages of FPGA include high flexibility, programmability, and low latency. It allows for rapid validation in the early stages of product development as it shortens the development cycle and enables real-time circuit design updates during production. It reduces the high costs associated with traditional chip development. Moreover, the parallel computing capabilities of FPGAs enable more efficient computations and the handling of intensive tasks, with hardware optimization according to specific applications. This also supports high-performance applications. These characteristics make FPGAs a good choice for high performance and flexibility requirements.

3. Proposed Architecture and Processing Procedures

For cost savings, real-time monitoring, lightweight design, and versatility in a tunnel traffic state monitoring system, the system developed in this study leverages the advantages

of FPGA hardware performance and the convenient features of the PYNQ-Z2 platform for visual computation. A lightweight intelligent traffic monitoring system is developed on top of existing monitoring systems, thereby reducing the system's construction costs. The developed system integrates image capture and real-time computation from roadside devices, enabling the monitoring of real-time mixed traffic and facilitating the intelligent management of the traffic system. The lightweight design algorithm performs real-time computations on FPGA, allowing for rapid synchronization of information to the management side and implementing AI applications for incident reporting, risk prediction, blind spot warnings, and event alerts within tunnel traffic.

The system development flow diagram is shown in Figure 1. It includes foreground vehicle tracking, background lane marking detection, and vehicle trajectory regression. Subsequently, each subsection introduces the relevant processes of foreground detection, vehicle detection, vehicle tracking, landmark detection, and training models.

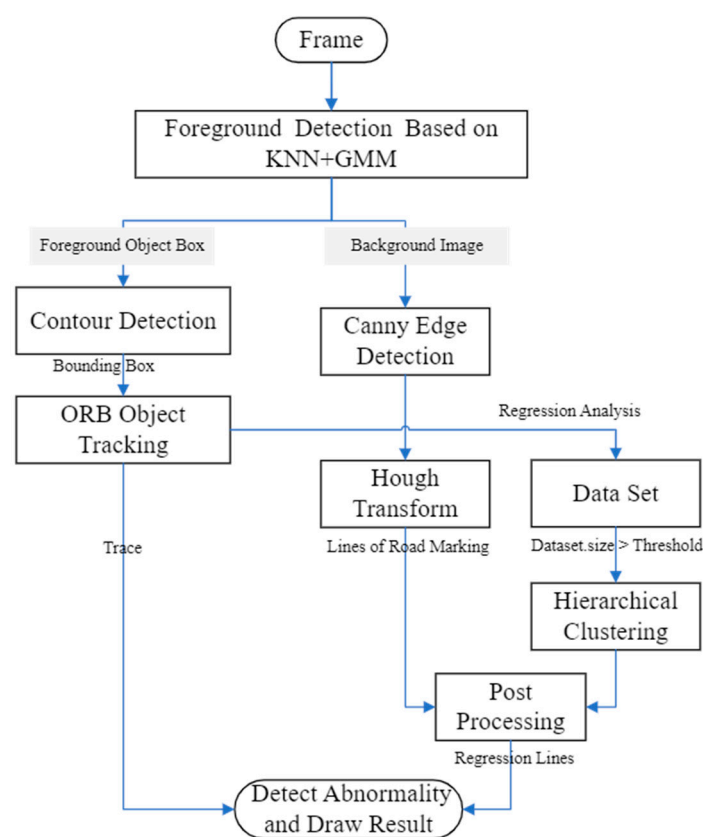


Figure 1. Flowchart of proposed architecture using visual computation and FPGA lightweight hardware acceleration.

3.1. Foreground Detection

The problem of technological law enforcement in tunnels is to detect abnormal situations, including unauthorized lane changes or traffic accidents. Real-time monitoring and abnormal detection are conducted through camera recordings, with the most direct method being the automatic acquisition of background images based on the recorded frame sequences. Due to differences in camera installation positions and variations in light sources, weather, and environments, the recorded frames also vary. To overcome this issue, we employed the KNN and GMM proposed by Zivkovic [4]. From the 400 frames recorded, 200 frames are processed to obtain the background image after removing those with significant differences. The foreground object frames are obtained by subtracting the

background image from the original sequence frames. The image in Figure 2a shows the traffic status in the tunnel, while Figure 2b displays the results of foreground detection.

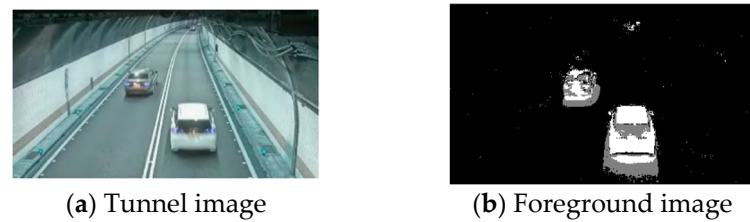


Figure 2. Foreground detection using KNN mixed GMM.

3.2. Vehicle Detection

The foreground sequence frames are used for real-time vehicle tracking and obtaining vehicle trajectories, while the background images are used for road sign detection. Based on the results of the foreground detection process, we employed the method by Suzuki and Keiichi [12] to perform contour detection as shown in Figure 2a. Figure 3 highlights the bounding boxes of the vehicles in motion.

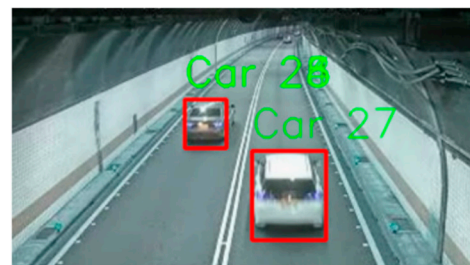


Figure 3. Vehicle detection based on foreground image from Figure 2b.

3.3. Vehicle Tracking

For vehicle tracking, we used the algorithm based on ORB features proposed by Wu et al. [8]. Their algorithm utilizes the object detection results to extract vehicle features to compare them with those of all detected objects in each frame, tracking the sequential positions of the same vehicle across different frames. The detected results of vehicle tracking under single and dual lanes are shown in Figure 4. Distinct colored boxes represent different cars.

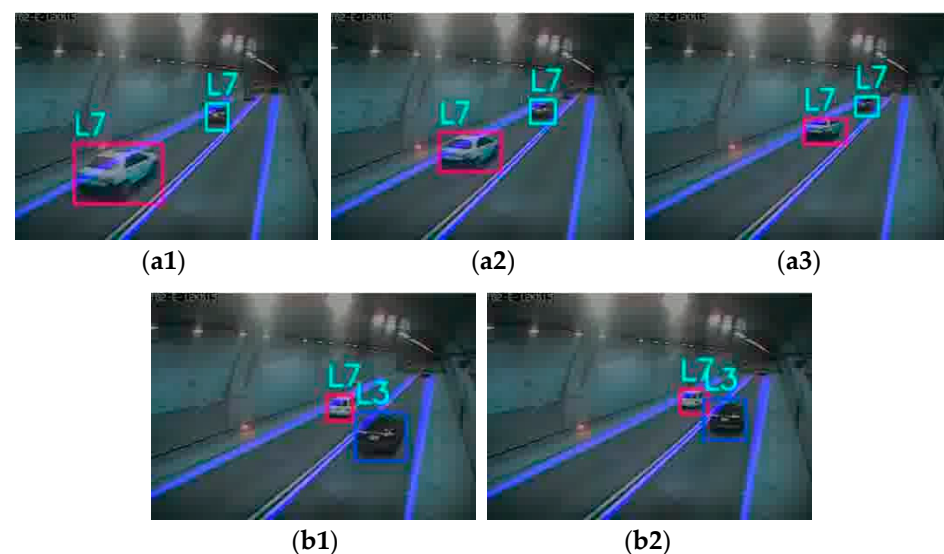


Figure 4. Vehicle tracking; (a1–a3) and (b1,b2) show the different lanes situations.

3.4. Road Marking Detection

We obtained the background image through model estimation from foreground detection. After filtering for specific Hue values, we applied the Canny algorithm [15] for edge detection and finally used the Hough transform [11] to obtain the regression lines of the road markings. Figure 5a shows the background image inside the tunnel, while Figure 5b shows the road markings represented by the red lines.

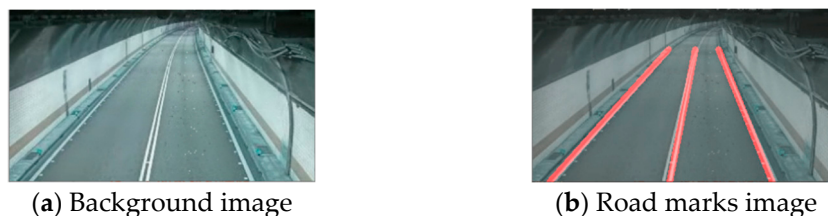


Figure 5. Line detection for road markings.

3.5. Training Model

We performed regression analysis on the vehicle tracking results to obtain the regression lines representing each vehicle's movement path with different colors. We then used hierarchical clustering [16] to group the paths and determine the regression lines and the number of lanes for each road. Finally, using the results from Road Marking Detection, we optimized the clustering results, as shown in Figure 6. This outcome serves as the typical driving path for most compliant vehicles.

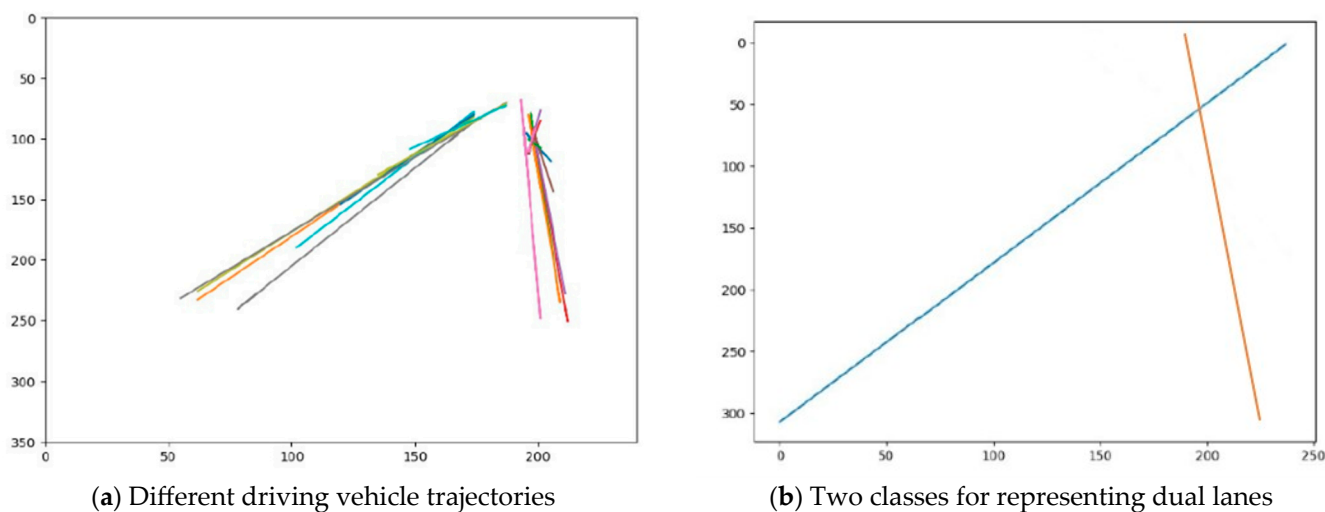


Figure 6. Landmark regression.

4. Results and Discussions

The developed system was implemented on a lightweight FPGA hardware platform that was integrated into the existing monitoring system. By doing this, costs are saved while preserving the original real-time monitoring capabilities. We conducted partial hardware acceleration and real-world testing. Furthermore, based on the proposed method, we present feasible solutions for future traffic law enforcement.

4.1. Hardware Acceleration

FPGA stores data within its internal hardware resources and directly transfers data between processing modules. FPGA reduces the frequency of memory access and latency, thereby further enhancing computational efficiency. These characteristics enable xfOpenCV

to compute with high efficiency and low power consumption on Xilinx's FPGA. Therefore, we utilized hardware-based image processing to accelerate the proposed vehicle tracking system. We adopted Xilinx's FPGA to customize hardware modules for specific image processing algorithms, achieving hardware acceleration [17]. By leveraging the customizable pipelined architecture of FPGA, different image processing stages are executed simultaneously. For foreground and road marking detection, we employed morphological operators such as dilation and erosion, which are well-suited for parallel processing to speed up preprocessing efficiency. Additionally, in road marking detection, we used Canny edge detection [15] to pre-identify the contours of road markings, and through parallel computation, we improved image processing speed. At the same time, by reducing the computational workload, we enhanced the efficiency of hardware acceleration.

4.2. Performance Test

The proposed combination of visual computing with lightweight FPGA on the PYNQ-Z2 platform for hardware acceleration in tunnel traffic tracking systems was compared with the conventional YOLO-v2-tiny [18], pre-trained on the common objects in context (COCO) dataset with 80 classes. The developed method significantly demonstrated a performance improvement. The comparison results are shown in Table 1.

Table 1. Table Type Styles.

	Developed Method	YOLO-v2-Tiny with Hardware Acceleration
Computing Speed	480p@20fps	480p@2fps
RAM	<1 MB	~127 Mb
Training Data	200 frames	$\sim 3 \times 1053 \times 105$ pictures
Training Time	<10 s	N/A

4.3. Traffic Enforcement

Technological law enforcement in traffic utilizes software and hardware technologies, such as AI, automatic detection devices, and wireless transmission, and assists police officers in managing traffic and reducing violations. This approach enhances enforcement efficiency and public safety. Through the system developed in this study, the following types of traffic law enforcement can be applied in the future.

- Vehicle trajectory detection: After detecting vehicles in each frame, vehicle tracking is enabled by analyzing image similarity, distance, and size, assigning a unique ID to each vehicle. The coordinates in each frame are then used for trajectory analysis.
- Lane-switching recognition: By collecting images and filtering out variations, the background view is obtained from the camera's perspective and analyzes the lane boundaries. With the integration of trajectory analysis, the system identifies lane-switching behaviors, including weaving or driving across lane markings.
- Illegal left and right turn detection: When the camera is positioned at an intersection, the system detects illegal turning behaviors through trajectory analysis and lane recognition.
- Traffic flow statistics: Whenever a new vehicle appears on the screen, it is counted, providing real-time statistics on the number of vehicles entering the camera's field of view per minute, enabling congestion analysis and monitoring.

5. Conclusions

To address the issue of traffic violation enforcement, we have designed a versatile and lightweight system based on the Xilinx FPGA platform PYNQ-Z2. This system has the following advantages:

- Real-time traffic monitoring in tunnels to instantly monitor lane violations and notify law enforcement agencies for subsequent handling.
- Collision prevention in tunnels to immediately alert vehicles behind to reroute and avoid rear-end collisions and traffic congestion in case of an accident.
- Reduced system size and cost by minimizing the system architecture, improving computational speed, and reducing system development costs.
- Automatic adaptation: Since it does not require pre-training, the system automatically adjusts to the local environment and enhances its versatility and ease of use.
- Seamless integration with existing cameras: The system shares and analyzes footage from existing surveillance systems without new installations, reducing deployment costs and increasing coverage of the intelligent monitoring system on the roads.

Finally, in terms of performance analysis, the developed system achieves a tenfold increase in computational speed compared to YOLO-v2-tiny for vehicle tracking. The memory consumption required for operation is less than 1% of the YOLO model's space usage.

Author Contributions: Conceptualization, Y.-C.L. and R.-S.L.; methodology, Y.-C.L.; validation, Y.-C.L. and R.-S.L.; formal analysis, Y.-C.L. and R.-S.L.; investigation, Y.-C.L.; resources, Y.-C.L.; data curation, R.-S.L.; writing—original draft preparation, Y.-C.L.; writing—review and editing, R.-S.L.; visualization, Y.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data are contained within the article.

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

References

1. Marusin, A.; Danilov, I. A method for assessing the influence of automated traffic enforcement system parameters on traffic safety. *Transp. Res. Procedia* **2018**, *36*, 500–506. [[CrossRef](#)]
2. Chen, L.; Zhu, Y. Impact of traffic enforcement technology on urban accident rates: A case study. *Int. J. Transp. Technol.* **2020**, *32*, 50–63.
3. Kumar, P.; Lee, R. Automation in traffic law enforcement: Benefits and challenges. *Transp. Res. Rev.* **2021**, *48*, 95–110.
4. Zivkovic, Z. Improved adaptive Gaussian mixture model for background subtraction. In Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, 26–26 August 2004; Volume 2, pp. 28–31.
5. Kim, J.B. Vehicle detection using deep learning technique in tunnel road environments. *Symmetry* **2020**, *12*, 2012. [[CrossRef](#)]
6. Ghanem, S.; Kanungo, P.; Panda, G.; Satapathy, S.C.; Sharma, R. Lane detection under artificial colored light in tunnels and on highways: An IoT-based framework for smart city infrastructure. *Complex Intell. Syst.* **2023**, *9*, 3601–3612. [[CrossRef](#)]
7. Sultana, S.; Ahmed, B.; Paul, M.; Islam, M.R.; Ahmad, S. Vision-based robust lane detection and tracking in challenging conditions. *IEEE Access* **2023**, *11*, 67938–67955. [[CrossRef](#)]
8. Wu, B.F.; Kao, C.C.; Liu, C.C.; Fan, C.J.; Chen, C.J. The vision-based vehicle detection and incident detection system in Hsueh-Shan tunnel. In Proceedings of the 2008 IEEE International Symposium on Industrial Electronics, Cambridge, UK, 30 June–2 July 2008; pp. 1394–1399.
9. Liu, W.; Wen, X.; Duan, B.; Yuan, H.; Wang, N. Rear vehicle detection and tracking for lane change assist. In Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 252–257.

10. Nguyen, V.Q.; Kim, H.; Jun, S.C.; Boo, K. A study on real-time detection method of lane and vehicle for lane change assistant system using vision system on highway. *Eng. Sci. Technol.* **2018**, *21*, 822–833. [[CrossRef](#)]
11. Duda, R.O.; Hart, P.E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM* **1972**, *15*, 11–15. [[CrossRef](#)]
12. Suzuki, S.; Keiichi, A. Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.* **1985**, *30*, 32–46. [[CrossRef](#)]
13. Wu, S.; Fan, Y.; Zheng, S.; Yang, H. Object tracking based on ORB and temporal-spatial constraint. In Proceedings of the 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), Nanjing, China, 18–20 October 2012; pp. 597–600.
14. Jadhav, S.; Narvekar, R.; Mandawale, A.; Elgandelwar, S. FPGA based object tracking system. In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 826–829.
15. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *6*, 679–698. [[CrossRef](#)]
16. Murtagh, F.; Contreras, P. Algorithms for hierarchical clustering: An overview. *WIREs Data Min. Knowl.* **2012**, *2*, 86–97. [[CrossRef](#)]
17. Zhai, J.; Li, B.; Lv, S.; Zhou, Q. FPGA-based vehicle detection and tracking accelerator. *Sensors* **2023**, *23*, 2208. [[CrossRef](#)] [[PubMed](#)]
18. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.