









Research Article

Utilizing Image Processing and the YOLOv3 Network for Real-Time Traffic Light Control

S. Francisco Segura Altamirano ¹, **Diana M. Castro Cárdenas** ¹,
Ayax M. Sifuentes Montes ², **Lucia I. Chaman Cabrera** ¹, **Esther Y. Lizana Puelles** ²,
Angel M. Rojas Coronel ³, **Oscar M. De la Cruz Rodríguez** ⁴, and **Luis A. Lara Romero** ⁵

¹Universidad Nacional Pedro Ruiz Gallo, Lambayeque, Peru

²Universidad Nacional de Piura, Piura, Peru

³Universidad Señor de Sipan, Chiclayo, Peru

⁴Antenor Orrego Private University, Trujillo, Peru

⁵National University of Trujillo, Trujillo, Peru

Correspondence should be addressed to S. Francisco Segura Altamirano; sseguraal@unprg.edu.pe

Received 21 February 2023; Revised 29 April 2023; Accepted 12 May 2023; Published 25 May 2023

Academic Editor: Ran Zhao

Copyright © 2023 S. Francisco Segura Altamirano et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this study, different strategies used to count vehicles and people in different image areas at a street intersection were analyzed to obtain counts at appropriate times suitable for real-time control of a traffic light. To achieve this, video recordings of cameras placed at the intersection were used to test and verify image processing algorithms and deep learning using the YOLOv3 network implemented on a 4 GB RAM Jetson Nano card. We counted the vehicles and people that stopped and crossed the polygons to delimit the different areas of interest, with a maximum error of ± 2 in the validation tests for all cases. In addition, as a strategy, we combined the images from both cameras into a single one, thereby allowing us to make a single detection and subsequently determine if they are inside or outside the polygons used in separating the areas of interest with the respective counts. Furthermore, this enabled us to obtain information on vehicles and people stopped and crossing in a time of 0.73 s on average. Hence, it was established that the inclusion of the control algorithm is appropriate for real-time control of traffic lights.

1. Introduction

In [1], a real-time traffic control system using various image-processing techniques collected using webcams was developed. The study used MATLAB software with timers to control the lights and a seven-segment display that shows the vehicle count. In addition, they used morphological image operations, but at the simulation level on flat surfaces with toy vehicles that are clearly different from those in the background. Furthermore, their study did not present results on real images or the time required by their algorithm. The study in [2] considered traffic congestion as a basic problem in urbanized areas. In their study, they used webcam images and MATLAB software to propose the control of traffic lights at a four-way intersection using RGB

conversion techniques and background subtraction with the objective of determining vehicle density, and 90% accuracy in density identification was achieved.

An investigation carried out on a toll road in [3] established that the traffic density can be controlled if the volume data of vehicles on the road are acquired and processed. Hence, they proposed a background subtraction method using a Raspberry Pi with OpenCV, which achieved a precision of 92.3% in the morning but dropped 77.3% in the afternoon, thus showing the dependency of the technique on lighting conditions, noise, vehicle speed, and camera viewing angle.

In [4], it was confirmed that traffic congestion is a significant problem occurring in all urban cities. The study proposed a vehicle detection model using image processing

by converting to RGB and HSV to determine whether the images are day or night. Furthermore, they applied the corresponding methodology to extract the vehicles and apply the object count, thereby achieving an average precision of 95% in the datasets used.

Furthermore, the study in [5] established that the inefficient control of traffic lights causes numerous problems, such as long delays and waste of energy. Therefore, using data collected from different sensors and vehicular networks, a reinforcement learning model was proposed to control the duration of traffic signals as actions that are modeled by Markov decision processes. To validate their model, they used the simulation of urban mobility (SUMO) environment, showing the efficiency of their model in controlling traffic lights.

In [6], we developed a new vehicle counting technique using a synergism attention network (SAN) to improve overlapping phenomena and sophisticated large-scale variations that occur in high-density images. They showed that the new SAN model obtains better performance indicators than MCNN, P2PNet, and DMCount.

All previous investigations consider congestion as a problem in cities, which can be confirmed by the data shown by the INRIX. Based on data collected from 300 million cars and connected devices at different times of the day along different paths of the road network, it was established that the cost of congestion in the United Kingdom was 37.7 billion pounds sterling [7], with an average of £1,168 per driver. Globally, trade increases as transport and logistics activities fulfil their functions, allowing the mobility of people and goods in an efficient, timely, and assertive manner. Furthermore, in the main cities of South America, there is up to 86 h lost in congestion, similar to that in São Paulo, thus representing an average of 30% of driving time lost in congestion. Hence, detailed knowledge of vehicular movement in transportation and logistics is necessary.

In addition, none of the previous studies provided information on the execution time of the proposed models or traffic light control algorithms. Hence, in this study, we propose the use of a YOLOv3 network for counting vehicles and people based on images obtained from cameras placed at an intersection. Furthermore, the method entails the use of a strategy that allows counting to be achieved in less than one second, thereby enabling its use in the control of traffic lights and improving traffic control in real-time.

As observed in previous research studies, none of them provide information regarding the execution times of the proposed models or traffic light control algorithms. Additionally, it is worth noting that several of these studies developed algorithms using MATLAB, which requires the use of a laptop or desktop computer and makes it unsuitable for real-world traffic light applications due to installation difficulties. Furthermore, some studies utilized OpenCV and single-board platforms but tested them in simulated scenarios with toy vehicles or virtual environments.

In this study, we propose the use of YOLOv3 for real-time counting of vehicles and people obtained from cameras placed at an intersection using a Jetson Nano board, making

it easier to implement in real-world situations. Thus, our research provides a more suitable and efficient strategy for real-time traffic light control compared to previous investigations.

2. Materials and Methods

Our interest is in digital images acquired by some digital cameras, which are discrete representations of processed data with spatial (arrangement) and intensity (color) information. In addition, we consider it as a multidimensional signal. The two discrete (2-D) dimensions represent the digital image, where $I(m, n)$ represents the intensity response of a sensor to a series of fixed positions ($m = 1, 2, \dots, M$); ($n = 1, 2, \dots, N$) in 2D Cartesian coordinates derived from a continuous 2D signal space $I(x, y)$, through a sampling process often referred to as discretization. The value of this position in 2-D is known as a pixel. An image contains one or more colors (channels) that define the color intensity at a particular pixel value at location (m, n) , which we denote as $I(m, n)$.

The image processing was performed automatically, following the steps of image acquisition and storage, pre-processing, segmentation, representation, description, and recognition interpretation [8]. We defined the actions of representation description with the use of convolutional networks [9]. In addition for recognition interpretation, we apply neural networks with deep learning [10].

A neural network (or neuron) functions (see Figure 1) by multiplying its inputs by their weights, adding with its polarization b , the net output s passed through a transfer function, and we show the sigmoid function, but other functions can be used, depending on the applications and learning algorithms [11].

Neural networks usually have more than one layer, and the connections between different neurons are the weights that are modified during training. In Figure 2, we have a set of weights in the first hidden and output layers and two layers of neural networks with weights that can be updated during training. Neural networks with more than one hidden layer are examples of nonlinear relationships that learn complex relationships with the ability to approximate continuous functions.

Convolutional neural networks (CNNs) are neural networks specially designed to work with data with spatially characteristics, such as images, and they are composed of convolutional layers that filter the input layers to find valuable features within the inputs [12]. In Figure 3, we apply a $7 \times 7 \times 3$ filter to a $38 \times 38 \times 3$ image to obtain a $32 \times 32 \times 1$ image. Hence, the dimensions have been reduced, but the image has more marked characteristics, which depend on the properties of the applied filter [13, 14].

The input images were split into $S \times S$ meshes, while the algorithm checks the bounding mesh frame. In addition, the deep-learning algorithm extracts the bounding boxes of each possible detected object. Furthermore, using an available set of inputs, deep learning was used to determine the object type in each box. Finally, this process was repeated for each box, with the objects being counted, and consequently returning a list containing the number of objects found [15].

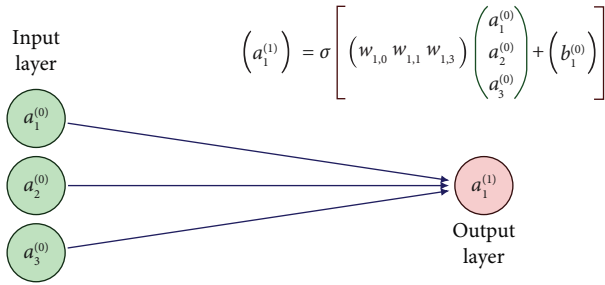


FIGURE 1: One-layer neural network.

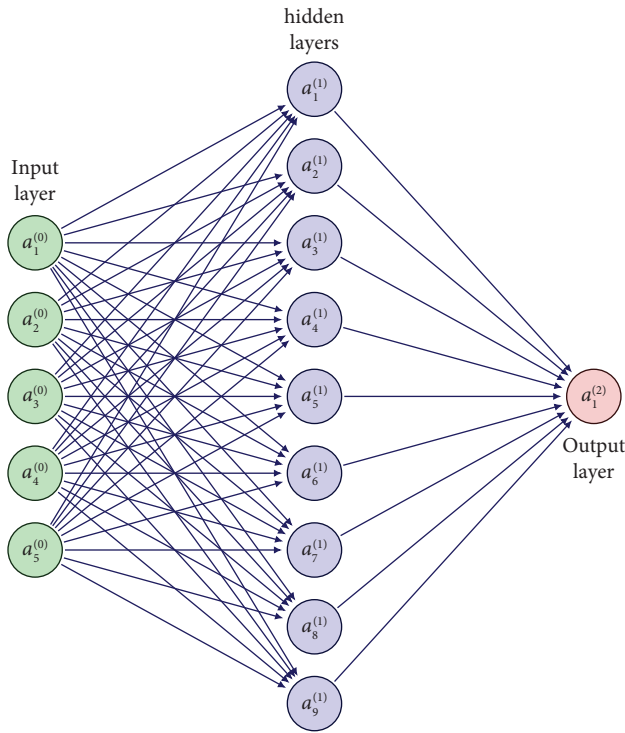


FIGURE 2: Neural network with a hidden layer.

The solution is built into an embedded device. Considering the processing speed and resolution of each frame, we reviewed the available embedded devices and selected the most suitable. The Raspberry Pi 3 B+ has 1 GB RAM, a 4-core processor with a 1.4 GHz clock frequency, four USB ports, an Ethernet port, and WiFi and Bluetooth connections [16]. The important feature of this Raspberry model is the availability of an HDMI port, a DPI camera port, and a display port. In addition, it was noticed that when AlexNet is used, a frame or video frame can be analyzed every 2.5 s, and in the worst case (BVLG reference models), an estimated time of 20 s fps. Furthermore, the Jetson Nano card is more powerful tool and can be used as a minicomputer alternative to the Raspberry Pi. It has 4 GB of RAM and 128 CUDA cores. Nvidia [17] showed the performance of this card with image recognition models, with speeds up to 25 fps, which is much higher than those of Raspberry Pi 3 B+ [17]. Conclusively, we compared the characteristics of both

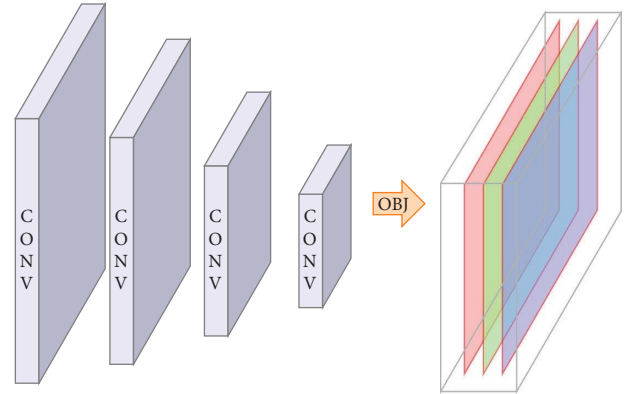


FIGURE 3: Convolutional networks.

minicomputers and performed simple detection algorithms on both platforms, hence we decided to work the solution based on Nvidia's Jetson Nano minicomputer.

YOLOv3 is pretrained network used for detect objects, and it consists of 106 layers and uses 80 categories (objects). For this purpose, the diagram was divided into $S \times S$ meshes. In each mesh box, the limits of the bounding boxes and a deep learning algorithm extract the bounding boxes for each detected object. In addition, the probability that each box contains an object of each category in the training set was calculated [18]. Furthermore, the obtained output consists of the location of the object given by the coordinates (x, y) , the size shown by $(W$ -width and H -height), and the probability of belonging to one of the 80 categories (objects).

2.1. Analysis and Proposal. We implemented of vehicle detection in videos and photos of the crossing of two one-way streets; by highlighting the precision of the YOLOv3 network (see Figure 4). First, we extracted relevant information about the detected objects (boxes of different colors in the image) and their quantities.

We analyzed the object detection performance on a set of images extracted from videos obtained from the crossing of two one-way streets, and made the following annotations (see Figure 5):

- (1) Vehicles before every intersection. Possibly stopped or what we see happening could be stopped by at color change
- (2) Vehicles are passing or in the transit zone
- (3) People are crossing the respective cross-walk
- (4) People are waiting on the corners for a possible change in the lights to be able to cross.

2.2. Detection with Masks. The first solution consists of filtering using a mask (see Figure 6) to obtain the corresponding counts. In total, we made four detections, two in each photo, to obtain, the number of stopped vehicles directly, people crossing, vehicles crossing, and people stopping. Therefore, the algorithm consists of applying Filter 01 to one of the streets, detecting and counting the stopped

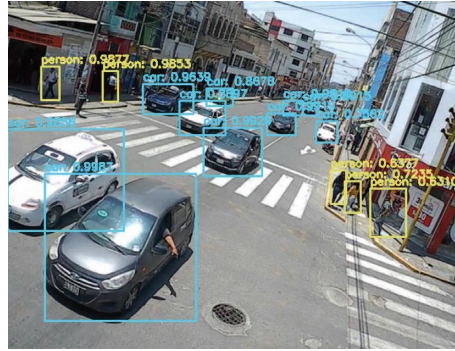


FIGURE 4: Detection of nine vehicles and five people.



FIGURE 5: Vehicle and person detection zones of the crossing of two one-way streets.

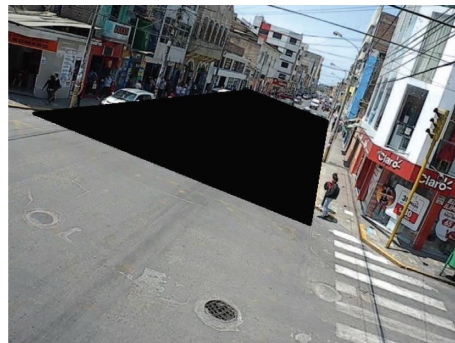


FIGURE 6: Zone of detection.

vehicles and crossing people in the resulting image, and subsequently using Filter 02 to see and count the passing vehicles and stopped people. This is repeated for the image from the other street, and then the traffic light logic is applied.

Furthermore, using a function from the time library and measuring the time with nanosecond precision, we determined the execution time of each detection process. In addition, we measured the total time and included the traffic light logic. It was found that, on average, each detection took about 0.7 s, with the duration of a complete cycle being 2.8 s.

2.3. Detection Using Polygon to Delimit Objects. Based on an aforementioned observation, the primary source of delay is the detection and counting processes in the different resulting images after applying the corresponding filters

(four detections and counts). Hence, we opted to perform only one detection in each image, thereby reducing the detections and counts to two (one for each image). In addition, we determined what we have detected (person or vehicle) and checking if they are inside or outside a polygon using a function `{point_in_polygon}`, with the ray-casting method [19, 20]. Hence, this allowed us to establish whether or not a given point is in an area delimited by a polygon. In addition, it enabled counting vehicles that stop and cross people in one case and vehicles that pass and people stop in the other. Thus, allowing the application of this information in the smart traffic light control logic.

Furthermore, we determined that the execution time of each detection process, in addition to each cycle of counting objects inside and outside the polygons in each image (on average 0.55 s, and the time counting objects and



FIGURE 7: Combined images of the intersection of one-way streets with polygons.

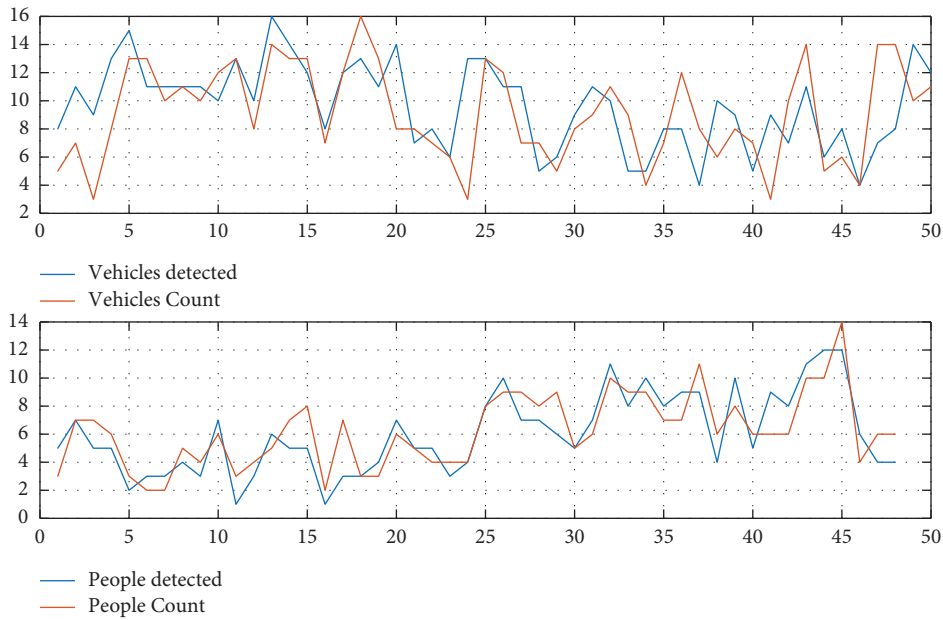


FIGURE 8: Vehicle and person detection and counting.

determining whether they are inside or outside the polygon, is less than 10 ms. Hence, completing the complete cycle per image in 1.14s considerably improves the performance mentioned previously.

2.4. Using Polygon to Clip Objects and a Single Detection. The detection and counting times exceeded one second. To reduce this total time, we chose to combine the images of both streets in a single shot (see Figure 7). In addition, we conducted a single detection process and used polygons that strengthen whether a given point is in an area to determine whether there are vehicles and people stopping or crossing each street.

3. Results and Discussion

In Figure 8, we compared the values obtained for the total number of vehicles and people in the images of crossing two one-way streets. Both statistics followed the same trend, and we confirmed the appropriate capability of the YOLOv3

network in detection and counting. In addition, we measured the performance using the mean error as follows:

$$e_m = \frac{\sum_{k=0}^L |V_d - V_c|}{L}, \quad (1)$$

where V_d values detected either vehicles or people; V_c values counted either vehicles or people; L number of evaluated images.

Applying this relationship, we obtained mean errors of 2.11 and 1.92 for vehicle count and people count, respectively. Thus, implying that on average, the results returned by the detector are in a range of \pm of the values found.

3.1. Discrete Counts. We carried out detection and counting using the zones of vehicles before each intersection and people crossing with described above (see Figure 5), with the results shown in Figure 9. The average errors in the vehicle and people counts are 1.85 and 1.03, respectively, thus,

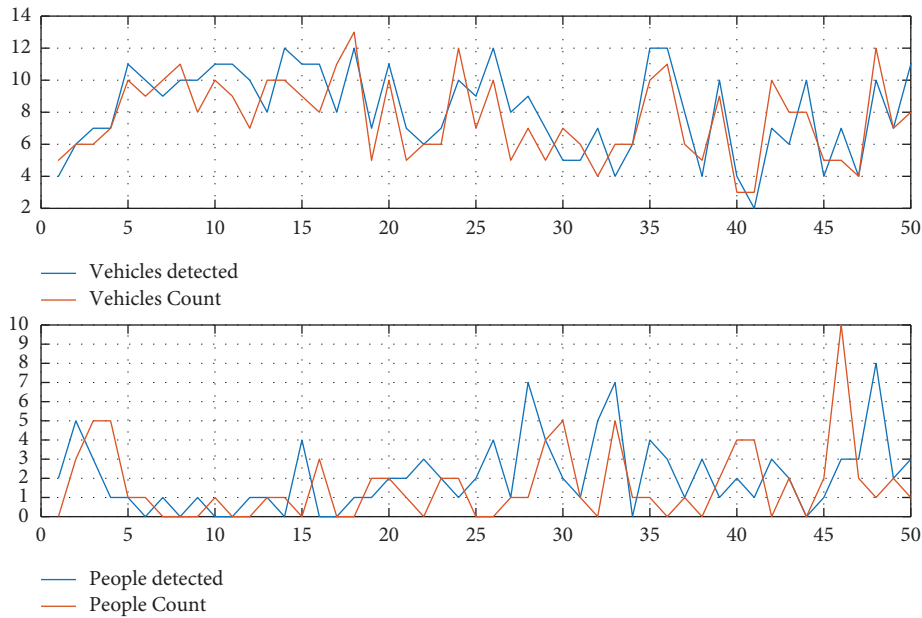


FIGURE 9: Vehicle and person detection and counting by zones.

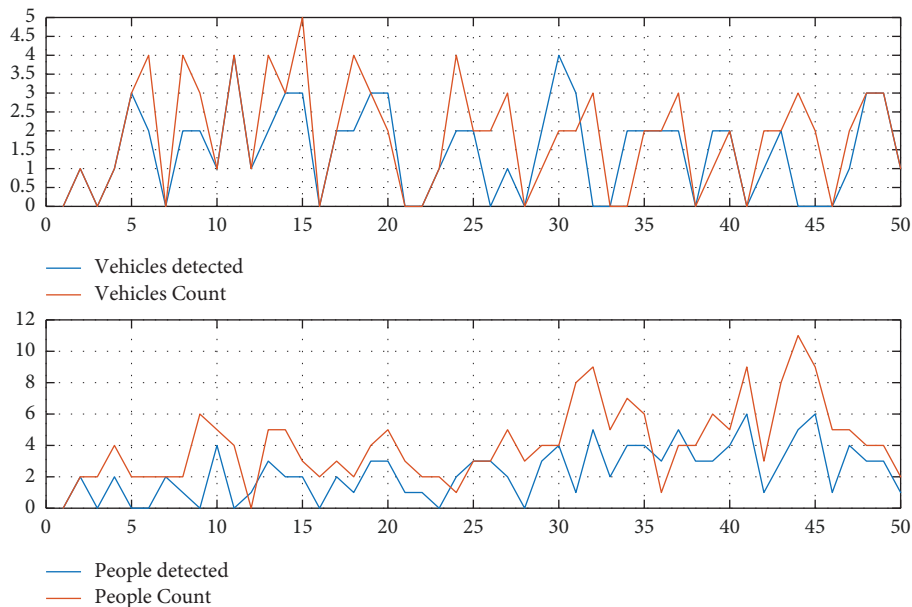


FIGURE 10: Vehicle and person detection and counting with mask.

implying that on average, the detector results can be within the range of ± 2 and ± 1 for vehicles and people, respectively.

We performed detection and counting in the zones shown in Figure 5, and the results are presented in Figure 10. It was found that the mean errors for the cases of vehicles and people are 2.07 and 1.11, respectively.

In addition, we determined the execution times of the unique detection process and the process of counting objects inside and outside each polygon for each street. The average detection time of the image was found to be 0.72 s. The time for measuring objects and determining if

they are inside or outside the polygon is less than 10 ms, while the time for completing the entire cycle per image is 0.73 s, thereby considerably improving the performance obtained. Furthermore, the use of other strategies was considered. The study of [2] proposes a traffic light control, using MATLAB, which is not appropriate for embedded devices. In addition to not indicating the execution time of the proposed control similar to [21], which uses CNNs to count objects in dense images, and does not present execution time results that allow its strategies to be applied to traffic light control.

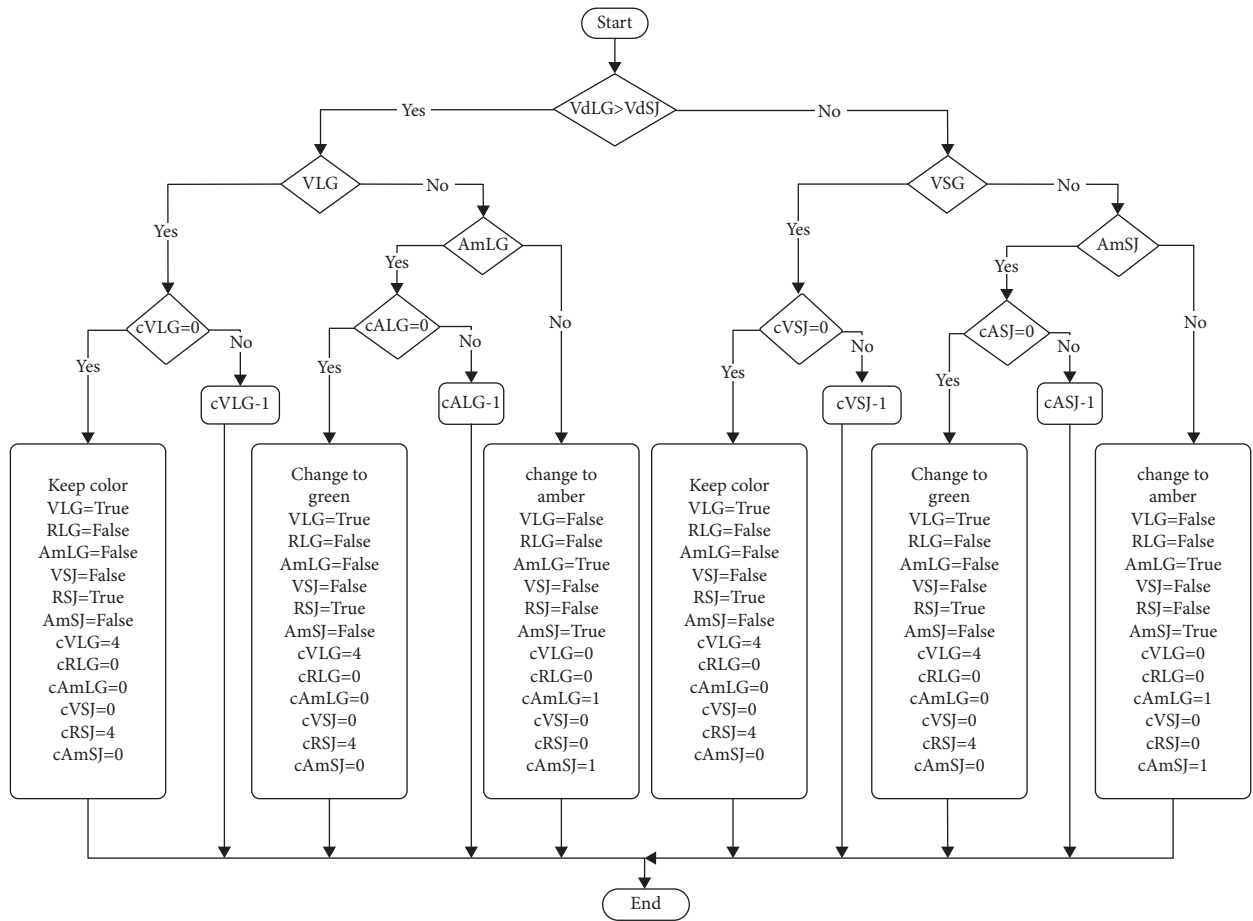


FIGURE 11: Real-time traffic light control.

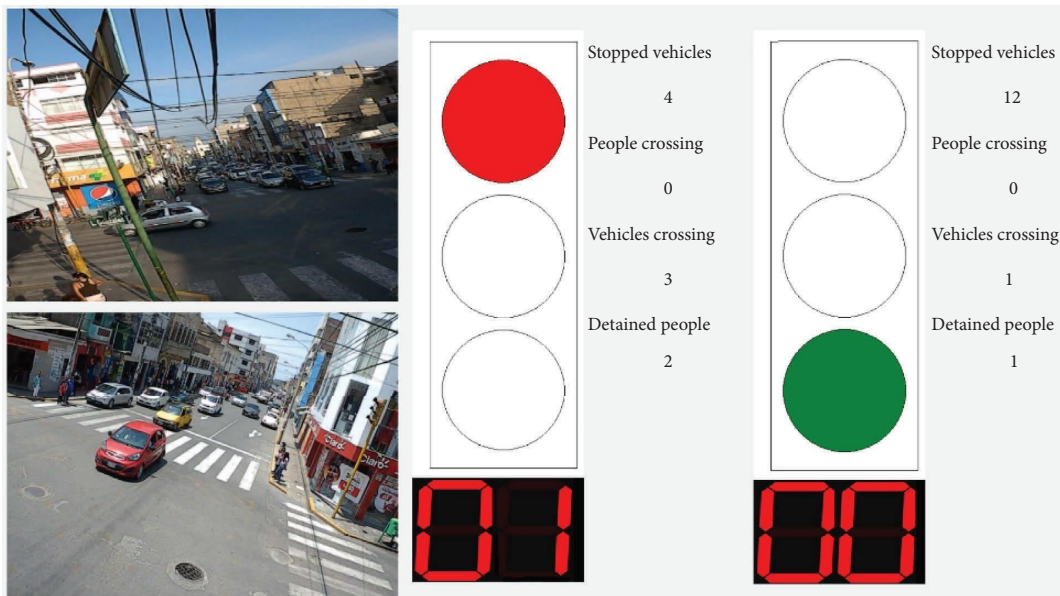


FIGURE 12: Test platform-GUI.

3.2. Real-Time Traffic Light Control. Based on the obtained times, which are less than one second in detection and counting, we developed a real-time traffic light control logic, which changes its state depending on the number of vehicles stopped at the intersection. Consequently, if there are a more significant number of vehicles stopped at one of the streets, this traffic light will turn green. To achieve this, it determines its current status. If it has been green, it simply does not make any change but decreases its counter by one. In addition, if it has not been green, it verifies the previous status, and if it has been amber and the amber counter is already at zero, it changes color to green and also updates the color of the traffic light that controls the other street by changing it to red, and further updates its new status and counters. Furthermore, if amber's counter was not zero, it decreased by one; if it is red, it makes a change again, passing through amber (see Figure 11).

We verified the functionality of the detection algorithms by testing a platform consisting of a graphic application, using simulated traffic-light control based on the counts obtained from photos or videos (See Figure 12).

4. Conclusions

This study aimed to analyze vehicular and pedestrian traffic at the crossings of two one-way streets and develop a real-time traffic light control system that considers the number of stopped vehicles, crossing vehicles, people crossing, and people stopping, with a minimum number of stopped cars and stopped people. The detector and counter algorithms used in the study showed promising results, with an average detection time of 0.72 s and a time for completing the entire cycle per image of 0.73 s, which is suitable for intelligent traffic light control.

The developed real-time traffic light control system can improve traffic flow and reduce congestion in urban areas, leading to more efficient and effective transportation and potentially reducing environmental impacts. The system can manage traffic efficiently and effectively by considering the number of stopped vehicles, crossing vehicles, people crossing, and people stopping, with a minimum number of stopped cars and stopped people. This study used a strategy of combining images from both cameras into a single shot, which improved the performance obtained.

However, the study used only two one-way streets, which may limit the applicability of the developed traffic light control system to other types of intersections or road configurations. Additionally, the performance of the developed system under real-world conditions is not tested, and this needs to be explored further.

Therefore, some future research directions based on the findings presented in the study could be exploring the performance of the developed real-time traffic light control system under real-world conditions to evaluate its effectiveness in managing traffic flow and reducing congestion in urban areas. Additionally, it could be extended to work with multiple intersections or road configurations to enhance its applicability and usefulness. Investigating the use of alternative object detection and counting methods or algorithms

to improve the performance and accuracy of the system and considering the integration of other sensors or data sources, such as GPS or weather data, could enhance the developed system's capabilities and effectiveness.

Data Availability

The corresponding author can provide the data used to support the findings of this study upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

S. Francisco Segura Altamirano and Diana M. Castro Cárdenas made substantial contributions to the conception or design of the work and approved the final version of the manuscript. Ayax M. Sifuentes Montes performed acquisition, performed analysis of data, and approved the final version of the manuscript. Lucia I. Chaman Cabrera and Esther Y. Lizana Puelles drafted the work or revised it critically for important intellectual content and approved the final version of the manuscript. Angel M. Rojas Coronel performed acquisition, performed analysis, interpreted data for the work, and approved the final version of the manuscript. Oscar M. De la Cruz Rodriguez provided agreement to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved and approved the final version of the manuscript. Luis A. Lara Romero provided agreement to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

- [1] M. H. Tunio, I. Memon, G. A. Mallah, N. A. Shaikh, R. A. Shaikh, and Y. Magsi, "Automation of traffic control system using image morphological operations," in *Proceedings of the 2020 International Conference on Information Science and Communication Technology (ICISCT)*, pp. 1–4, KARACHI, Pakistan, Feb, 2020.
- [2] S. Balu and C. Priyadharsini, "Smart traffic congestion control system," in *Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 689–692, IEEE, Erode, India, March, 2019.
- [3] H. Yuliandoko, M. D. Ayatullah, and F. E. Purwita, "Automatic vehicle counting using Raspberry pi and background subtractions method in the sidoarjo toll road," in *Proceedings of the 2019 2nd International Conference of Computer and Informatics Engineering (IC2IE)*, pp. 83–86, Banyuwangi, Indonesia, September 2019.
- [4] P. N. Chowdhury, T. Chandra Ray, and J. Uddin, "A vehicle detection technique for traffic management using image processing," in *Proceedings of the 2018 International Conference on Computer, Communication, Chemical*, pp. 1–4,

- Material and Electronic Engineering (IC4ME2), Rajshahi Bangalae dh, February 2018.
- [5] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, Feb, 2019.
- [6] Y. Jin, J. Wu, W. Wang, Y. Wang, X. Yang, and J. Zheng, "Dense vehicle counting estimation via a synergism attention network," *Electronics*, vol. 11, pp. 3792–3822, 2022.
- [7] P. Inrix, *Traffic Congestion Cost UK Motorists over £37.7 Billion in 2017*, Inrix Kirkland, Washington, DC, USA, 2018.
- [8] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, WILEY-BLACKWELL, 1 edition, 2011, <https://www.wiley.com/en-us/Fundamentals+of+Digital+Image+Processing%3A+A+Practical+Approach+with+Examples+in+Matlab-p-9780470844724>, Hoboken, NY, USA.
- [9] D. Sandipan, *Hnads-On Image Processing with Python*, Packt Publish, Birmingham, 1 edition, 2018.
- [10] C. Szegedy, A. Toshev, and D. Erhan, *Deep Neural Networks for Object Detection*, NeurIPS, Tahoe, 2013.
- [11] F. Chollet, *Deep Learning with Python*, Manning Publications, Shelter Insland, NY, USA, 1 edition, 2017.
- [12] R. A. Hadi, G. Sulong, and L. E. George, "Vehicle detection and tracking techniques a concise review," *Signal and Image Processing International Journal*, vol. 5, no. 1, pp. 1–12, 2014.
- [13] M. P., G. M. K. Krishna, A. S. Prabhu, and V. Umadevi, "Automated traffic monitoring system using computer vision," in *Proceedings of the 2016 International Conference on ICT in Business Industry and Government (ICTBIG)*, pp. 1–5, Indore, India, November 2016.
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
- [15] W. Zhao, S. Du, and W. J. Emery, "Object-based convolutional neural network for high-resolution imagery classification," *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 7, pp. 3386–3396, 2017.
- [16] R. Ayachi, M. Afif, Y. Said, and A. Ben Abdelali, "An edge implementation of a traffic sign detection system for Advanced driver Assistance Systems," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 2, pp. 207–215, 2022.
- [17] D. Nvidia, "Jetson Nano: deep learning inference benchmarks," 2019, <https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks>.
- [18] L. de Oliveira, L. T. Manera, and P. D. G. D. Luz, "Development of a smart traffic light control system with real-time monitoring," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3384–3393, Mar, 2021.
- [19] R. Li, T. Huang, X. Zhang, and H. Liao, "4: interactive volume rendering method using dynamic ray casting for autostereoscopic display," *SID Symposium Digest of Technical Papers*, vol. 52, no. 2, pp. 26–29, 2021.
- [20] G. Paulin, S. Sambolek, and M. Ivasic-Kos, "Person localization and distance determination using the raycast method," in *Proceedings of the 2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–5, Split, Croatia, September 2021.
- [21] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 833–841, Boston, MA, USA, June 2015.