Journal of Al-Azhar University Engineering Sector

Vol. 19, No. 73, October 2024, 1382 - 1395

AUTOMATED VEHICLE COUNTING AND SPEED ESTIMATION USING YOLOV8 AND COMPUTER VISION

Sherif A. Kamal[*1], **Ahmed Gaber**[2], **Mohammad A. Shehata**[2]

[1] Construction and Building Department, October 6 University, Giza 12511, Egypt

[2] Geology Department, Faculty of Science, Port Said University, Port Said 42522, Egypt

***Correspondence:** sherif.ahmad.eng@o6u.edu.eg

Citation:
S. A. Kamal, A. Gaber, M. A. Shehata, " Automated Vehicle Counting and Speed Estimation Using Yolov8 and Computer Vision ", Journal of Al-Azhar University Engineering Sector, vol. 19, pp. 1513 - 1526, 2024.

Received: 18 April 2024

Revised: 28 July 2024

Accepted: 08 August 2024

DOI:10.21608/auej.2024.280422.1642

ABSTRACT

Rapid urbanization requires very effective roadway planning to overcome traffic congestion problems. The traffic density can be monitored with manual counting or advanced instruments. However, these conventional techniques are complicated, costly, and often inaccurate. On the other hand, automized methods using video data and AI techniques provide a cost-effective method that can measure vehicle counts and speed with adequate accuracy. This paper provides a Python program for vehicle counting from recorded videos using the YOLOv8 algorithm and Computer Vision (CV). The program has been tested for three recorded videos from three different roads at different times of the day. The manual counts of the number of vehicles were recorded to assess the program's efficiency. Results showed that the program achieved an accuracy rate of 96% compared to the manual counts. This flexible program can be modified to work with streaming videos, providing real-time vehicle counting and enabling traffic congestion prediction.

KEYWORDS: Vehicle counting; Traffic counting; AI Deep Learning

التعداد الآلي للمركبات و تقدير السرعات من خلال الذكاء الاصطناعي

شريف احمد مصطفى [1*]،احمد جابر[2]، محمد شحاتة [2]

[1] قسم التشييد و البناء ، جامعة ٦ أكتوبر، الجيزة ١٢٥١١، مصر

[2] قسم الجيولوجيا، كلية العلوم، جامعة بورسعيد، بورسعيد ٤٢٥٢٢، مصر

***البريد الاليكتروني للباحث الرئيسي :** sherif.ahmad.eng@o6u.edu.eg

الملخص

يتطلب التوسع الحضري السريع تخطيطًا فعالًا للطرق لتجاوز مشاكل الازدحام المروري. يمكن رصد كثافة المرور عن طريق العد اليدوي أو الأجهزة المتقدمة. ومع ذلك، فإن هذه التقنيات التقليدية معقدة ومكلفة وغير دقيقة في كثير من الأحيان. من ناحية أخرى، توفر الطرق الآلية باستخدام بيانات الفيديو وتقنيات الذكاء الاصطناعي طريقة فعالة من حيث التكلفة يمكنها قياس عدد المركبات والسرعة بدقة كافية. تقدم هذه الورقة برنامج بايثون لعد المركبات من مقاطع الفيديو المسجلة باستخدام خوارزمية YOLOv8 ورؤية الكمبيوتر (CV). تم اختبار البرنامج على ثلاثة مقاطع فيديو مسجلة من ثلاثة طرق مختلفة في أوقات مختلفة من اليوم. تم تسجيل عدد المركبات يدويًا لتقييم كفاءة البرنامج. أظهرت النتائج أن البرنامج حقق معدل دقة بنسبة 96٪ مقارنة بالعد اليدوي. يمكن تعديل هذا البرنامج المرن للعمل مع مقاطع الفيديو المتدفقة، مما يوفر عد المركبات في الوقت الفعلي ويمكّن من التنبؤ بالازدحام المروري.

الكلمات المفتاحية : عد المركبات، حصر مرورى، التعلم العميق بالذكاء الاصطناعي.

1. INTRODUCTION

In situ technology and computer algorithms are two of the many ways that autonomous vehicle surveillance may be accomplished. In situ approaches use roadside monitors to measure traffic volume accurately. Often, Sensors include piezoelectric sensors, microwave radar, magnetic loops, and pneumatic road tubes. Fog, rain, wind, traffic volume, and road surface are some of the climatic elements that impact the accuracy of these expensive technologies. Using machine learning algorithms, which automatically determine traffic volume by evaluating past camera records, is an enhanced alternative to conventional methods.

Many well-established frameworks are available for selection when developing algorithms for computer-based traffic measurement. Image processing technologies are essential for all models' approaches [1]. The sequential Monte Carlo technique and the traditional background reduction procedures are widely used in image processing. The background subtraction technique may be used to separate the mobile component of the picture from the surrounding frame. One of the several potential methods for reducing background is to create a model for each pixel in the image. [2] used a Kalman filter to simulate individual pixels in the background and differentiate them from the ones that were not.

On the other hand, [3] used a single Gaussian value to determine the likelihood of a pixel being in the background. In the early days of backdrop suppression algorithms, problems were shared, including slow cars, poor illumination, and shaded areas. Also, most of these systems can't handle actual car numbers because of their slow video processing speeds.

To address these issues, [4] put out a technique wherein a composite of Gaussian values was used to represent each pixel. To use the method, one must divide each frame into several pixels and then express each pixel by established rules. An adaptable Gaussian is used to estimate the lighting qualities of many surfaces and different lighting circumstances, in contrast to a single Gaussian that represents the lighting characteristics of a pixel with a single surface. Object recognition is accomplished by Clustering Gaussian values distinct from those in the preceding frame using connected components. Observing moving objects frame by frame allows one to determine their direction of motion. The approach is slow and inefficient when dealing with big, overlapping objects; it only manages to classify them, not identify them.

[5] developed a state-of-the-art approach that used rule-based reasoning and image processing to track vehicles in videos. All hours of the day and night may be monitored using the aforementioned technology. During the day, the spatiotemporal module finds and follows clusters of moving pixels from one frame to the next. Their technology can detect and identify moving particles by evaluating three successive frames. The number of cars is calculated using a complex knowledge-based technique that accurately differentiates moving, stopped, and crossing vehicles. The method can detect cars with two headlights by morphologically examining nightlights. The program separates the two headlights from the rest of the picture using image-masking and thresholding methods. Headlight pairings that match are recognized as automobiles using a template comparison method. Its inability to discern vehicles with a single headlight reduces the approach's usefulness in identifying motorbikes or cars.

[6] used a sequential Monte Carlo approach to detect, track, and verify items. There are three distinct phases to the procedure. Its density, position, and speed define its present condition. [7] also describe how items inside a specific frame are identified using the sequential significance sampling (SIS) approach. By entering a particular parameterization, such as the object's location,

into the state 'x' during the first phase, the object's identification as a target object is established, and its trajectory is traced between frames. The results show that the algorithm provides a valuable approach to observing and verifying objects.

The researchers faced a significant challenge in automated traffic surveillance while trying to identify items in a picture and remove unwanted shadows properly. [8] used the novel line-based shadow strategy to tackle these problems in their research. Vehicles are first separated from their backgrounds using image differencing. The next step is to use a shadow-removal technique to make these automobiles less noticeable. When using a Kalman filter for vehicle tracking, it is necessary to assess the position and speed of the vehicles. Connected component analysis and line fitting are used to determine the vehicle dimensions. Vehicles are classified according to the two characteristics described above. When it comes to tracking and classifying cars, this system works effectively.

Several object identification algorithms, including You Only Look Once (YOLO), Single Shot Multi-Box Detector (SSD), Fast R-CNN, and Region-Based Convolutional Neural Networks (R-CNN), have been created due to improvements in deep learning. The R-CNN and Fast R-CNN algorithms, developed by [9], use specific queries to detect objects. [10] created a method for object detection dubbed the Single Shot Detector (SSD). The technique employs a grid-based strategy to partition a picture into smaller segments. The goal of each grid cell is to detect and recognize objects inside its defined zone. [11] developed a computer technique, YOLO, to identify objects. Only a single network loop is required to analyze an image and identify objects. [12] have shown that the YOLO model exhibits superior speed and accuracy compared to earlier models. YOLO employs the non-maximal suppression technique in image processing, which surpasses traditional detectors in terms of effectiveness. The YOLO method starts by partitioning the input picture into a grid of cells 13x13. Each of these cells is responsible for predicting numerous image boundary boxes. YOLO algorithms calculate a singular probability value for every bounding box. A threshold value is used to ascertain the suitability of a bounding box for object identification.

This research aims to provide an efficient object detection model for cost, time, and accuracy by utilizing the YOLO object detection model.

2. MATERIALS AND METHODS

2.1 Video Data

Traffic videos from the Cairo-Suez Road, Ring Road Ismailia, and Port Said entrance were acquired. The video is HD and six hours long, in mp4 format. Both directions of the road were recorded in this video.

2.2 Yolo Pre-Trained Model.

YOLO version 8 was selected for this investigation. This version exhibits much-improved speed, accuracy, and user-friendliness. The YOLO version 8 algorithm analyzes pictures at a resolution of 640x640 pixels. Version 8 has a superior resolution, allowing it to discern even the most minute elements in photographs more precisely than its predecessors. The outcomes of implementing the suggested approach using YOLO version 8 remain unaffected by the number of cars observed since it does not include identifying minuscule objects. The YOLO model was used for vehicle identification and quantization using the TensorFlow and OpenCV libraries. All the algorithms were developed using the Python programming language.

2.3 TensorFlow API Preparation

The original code for YOLO weight files was written in C/C++. A TensorFlow structure was created from the weight file for Python use. This transformation could not have been accomplished without the following: Python, the open-source computer vision library OpenCV, the NumPy module for mathematical operations in Python, the TensorFlow 18.0 framework, and TQDM, a package for showing progress bars in nested loops. The YOLO weight file was obtained from the official repository of the YOLO website. The YOLO weight file has been formatted using TensorFlow.

2.4 Input Files

This stage sets up the directory where the YOLO weight file, anchor definitions, YOLO model class definition, and the number of available graphics processing units are stored. The input pre-recorded video name and directory arguments were generated by the 'parser' job. The arguments were created and carried out in the 'main' function, which oversees the vehicle identification algorithms.

Video files must be processed before they can be submitted to the application. Transcode all video files to the mp4 format. For efficient uploading of several video files in the software, it is advisable to consolidate them into a single file, as the application cannot upload multiple movies simultaneously. The video filename is input manually before the '.mp4' extension.

2.5 Vehicles Detection

Algorithms were created to establish reference lines and identify automobiles in every video frame. Using these methods, users may create a reference line by dragging the mouse pointer over the first video frame. The reference lines validate the program's accuracy in obtaining an accurate automobile count in subsequent stages. The cars in every video frame were identified using the YOLO weight file. Various methods have been devised to optimize this process, such as sizing every picture to a consistent size and setting a threshold to identify things that are likely to be recognized. The whole vehicle detection procedure is shown in Figure 1.

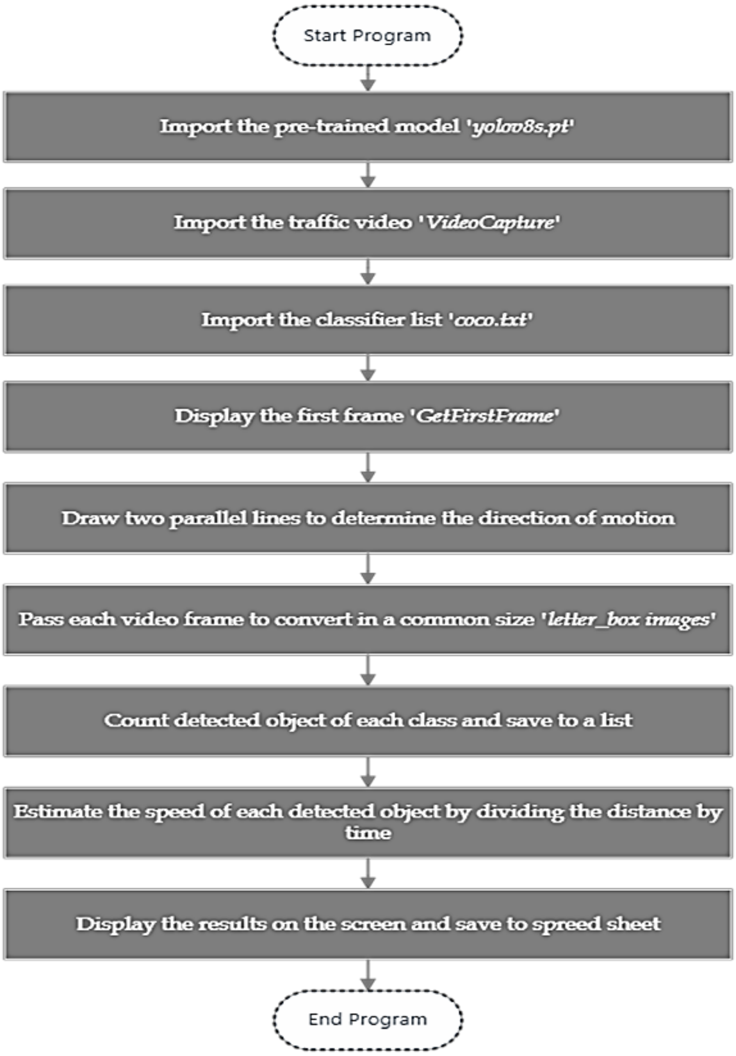

Start Program
Import the pre-trained model 'yolov8s.pt'
Import the traffic video 'VideoCapture'
Import the classifier list 'coco.txt'
Display the first frame 'GetFirstFrame'
Draw two parallel lines to determine the direction of motion
Pass each video frame to convert in a common size 'letter_box images'
Count detected object of each class and save to a list
Estimate the speed of each detected object by dividing the distance by time
Display the results on the screen and save to spreed sheet
End Program

Figure 1. The schematic flow diagram for a practical vehicle detection system.

The program invokes the 'detect_start' function and returns the file information if the input parameter contains the video file's name and directory. As shown in Figure 2, the application invokes the 'getFirstFrame' function to capture and display the initial frame using OpenCV. The algorithm invokes the mouse handling function 'setMouseCallback,' which enables the user to manipulate the cursor to determine the beginning and end points of a reference line (Figure 2).

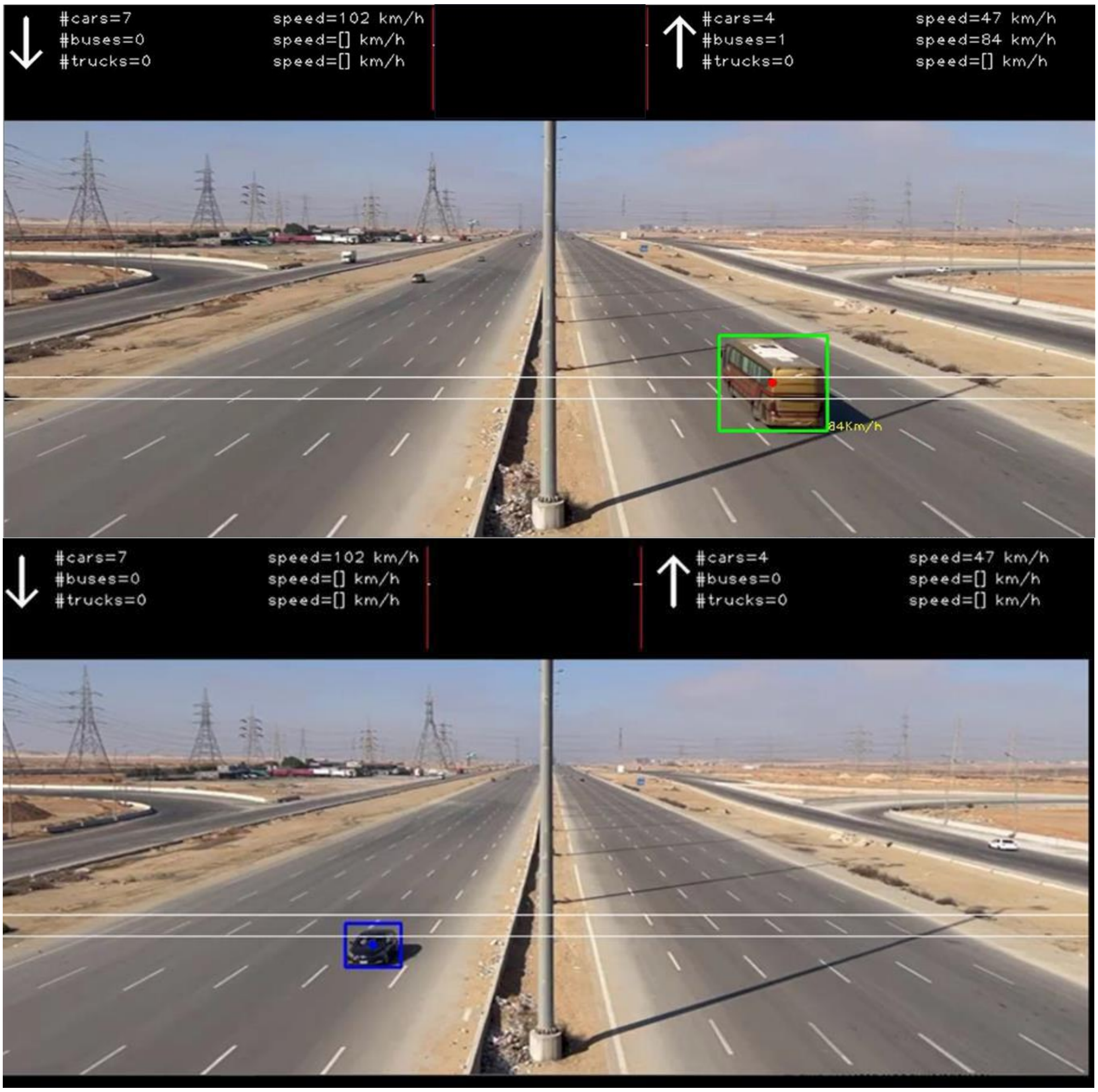


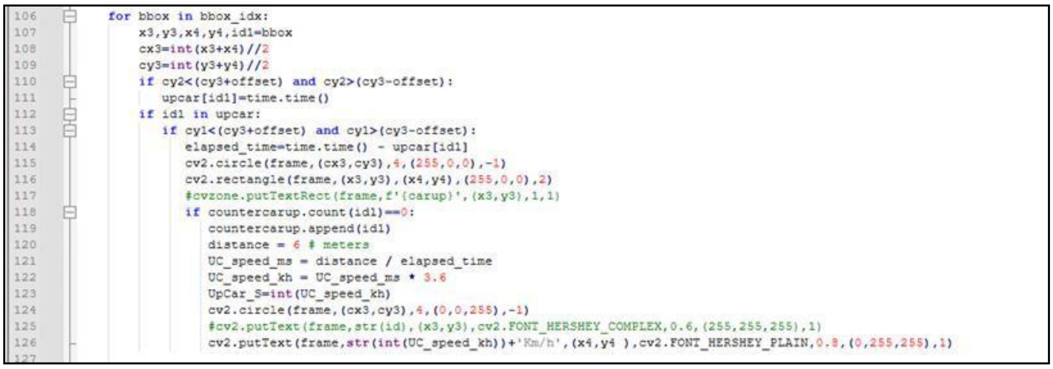
Figure 2. The user-drawn figure represents the frame, including its center point (represented by a blue dot) and two parallel lines in white.

The software then proceeds to execute the 'detect_image' method. This technique requires a single video frame and reference lines as parameters. Further functions defined within this function are 'get_right_line' and 'get_left_line,' both responsible for drawing parallel lines, as depicted by the white lines in Figure 2.

Each video frame is passed through the 'detect_image' method to the 'letter_box image' function. In conformity with the YOLO weight file provided by the TensorFlow API, every video frame is resized to a standard dimension of 640 × 640 pixels. TensorFlow detects objects within each frame utilizing the modified YOLO weight file. The YOLO weight file can currently identify every potential object in an image. After identifying an object in an image, YOLO produces a rectangular bounding box encircling it; the hue of this bounding box differs for each vehicle type. The image that has been processed produces rectangular bounding boxes, with each box being assigned a score. The confidence level in identifying an object is denoted by the score value, calculated on a scale from 0 to 1. YOLO utilizes non-maximal suppression to manage the bounding boundaries. A critical determinant in this procedure is a threshold value that excludes bounding boxes exhibiting low confidence. A value of 0.2 was employed as the threshold in the study. The following bounding boxes are not chosen for further processing: only those with a score above 0.2 are chosen. The structured bounding frames are transferred to the 'dets' array to monitor identified objects.

2.6 Vehicles Tracking

To determine the movement direction of an item in video footage, it must be tracked from one frame to the next since video is composed of several frames. The main goal of this section is to develop methods for consistently following the identified object through consecutive frames. The 'KalmanBoxTracker' monitors the organized bounding boxes from the 'dets' array by analyzing the pixel variance between the current and previous frames. Upon detecting a match in pixel values, the system modifies the object's bounding box, saves it for later use, and proceeds with the process. The tracker assigns numerical values to each bounding box in the investigation. Using the 'draw. Rectangle' technique was developed in the program, and a rectangle was generated around each detected object (Figure 3). The rectangular boxes are designed to have a dotted line drawn in their center to assist in counting autos, as elaborated in the section on counting vehicles. The dot marker indicates the centerline of each object.



```
for bbox in bbox_idx:
    x3,y3,x4,y4,id1=bbox
    cx3=int(x3+x4)//2
    cy3=int(y3+y4)//2
    if cy2<(cy3+offset) and cy2>(cy3-offset):
       upcar[id1]=time.time()
    if id1 in upcar:
       if cy1<(cy3+offset) and cy1>(cy3-offset):
          elapsed_time=time.time() - upcar[id1]
          cv2.circle(frame,(cx3,cy3),4,(255,0,0),-1)
          cv2.rectangle(frame,(x3,y3),(x4,y4),(255,0,0),2)
          #cvzone.putTextRect(frame,f'{carup}',(x3,y3),1,1)
          if countercarup.count(id1)==0:
             countercarup.append(id1)
             distance = 6 # meters
             UC_speed_ms = distance / elapsed_time
             UC_speed_kh = UC_speed_ms * 3.6
             UpCar_S=int(UC_speed_kh)
             cv2.circle(frame,(cx3,cy3),4,(0,0,255),-1)
             #cv2.putText(frame,str(id),(x3,y3),cv2.FONT_HERSHEY_COMPLEX,0.6,(255,255,255),1)
             cv2.putText(frame,str(int(UC_speed_kh))+'Km/h',(x4,y4 ),cv2.FONT_HERSHEY_PLAIN,0.8,(0,255,255),1)
```

Figure 3. The program assigns an ID for each detected item and draws a rectangle around it.

To define the direction of each vehicle, a conditional statement is added to monitor which line (from the white lines) is first crossed. The detected objects are stored in a dictionary with a specific ID. The speed of each object is then estimated using the time difference between the frames and the distance between the two white lines. The speed is also stored in the dictionary along with the id. A counter function counts the number of cars. Thus, the dictionary stores the ID, the number of objects, and their speed.

2.7 OpenCV and CUDA

Before running the program, it is necessary to utilize the YOLO model with OpenCV, an open-source computer vision library, to enable its functionality in a GPU context and attain faster video processing. To successfully finish this assignment, you will need two open-source software packages: cuDNN and CUDA. cuDNN is a GPU-accelerated library that is built on deep neural networks. CUDA is a programming language developed by NVIDIA specifically for coding graphics hardware. Correct installation of CUDA is crucial for effectively integrating YOLO with OpenCV and cuDNN.

2.8 Hardware Platforms

The speed at which the program counts depend on the hardware platform's settings. This investigation employed the subsequent hardware platform:
Processor: The computer is equipped with an Intel(R) Core (TM)i5-8250CPU that operates at a base frequency of 1.6 GHz and can reach a maximum turbo frequency of 1.8 GHz. RAM: The computer has 12.0 GB of memory that operates at a speed of 2.4 GHz. GPU: The computer has an NVIDIA GeForce GTX 1060 graphics card with 6 GB of dedicated video memory.
It was found on this platform that the program takes around one hour and forty to fifty minutes to process one hour of video. Using a computer with high-performance specifications is recommended for faster video processing. To expedite the processing time, it is recommended to utilize a graphics processing unit (GPU) with a video memory capacity exceeding 6 gigabytes.

3. RESULTS AND DISCUSSION

3.1 Automated Vehicles Counting

Pre-recorded videos were subjected to analysis utilizing computer algorithms that were developed. The duration of the video capture was six hours. The automated counting method was employed to compile the entry and exit counts for the recorded region, which are displayed in Table 1.

Table 1: Automated vehicle counting results.

Site	Downward direction	Upward direction
Cairo-Suez road	352	308
Ring Road Ismailia	148	173
Port Said Entrance	297	216

3.2 Accuracy Evaluation

The accuracy of the automated counting was evaluated by comparing it to manual counting. Table 2 presents the exact number of entries, exits, and overall counts.

Table 2: Accuracy of automated vehicle counts.

Site	Downward direction			Upward-direction			Total		
	Manual	**Automate d**	**Accuracy %**	**Manual**	**Automate d**	**Accuracy %**	**Manual**	**Automate d**	**Accuracy %**
Cairo-Suez road	352	339	96.3	308	298	96.7	637	660	96.5
Ring Road Ismailia	156	148	94.8	182	173	95.1	338	321	94.9
Port Said Entrance	309	297	96.1	222	216	97.3	531	513	96.6

3.3 Paired t-test

The two-tailed paired t-test is a reliable statistical technique used to evaluate the disparities between the measurements of two variables. The similarity between human and computer counts was assessed using a two-tailed paired t-test. This investigation entailed performing counts at 10 locations using both automated and manual approaches. In this situation, the locations are the same subjects of interest, whereas the human and automated counts are considered separate variables.

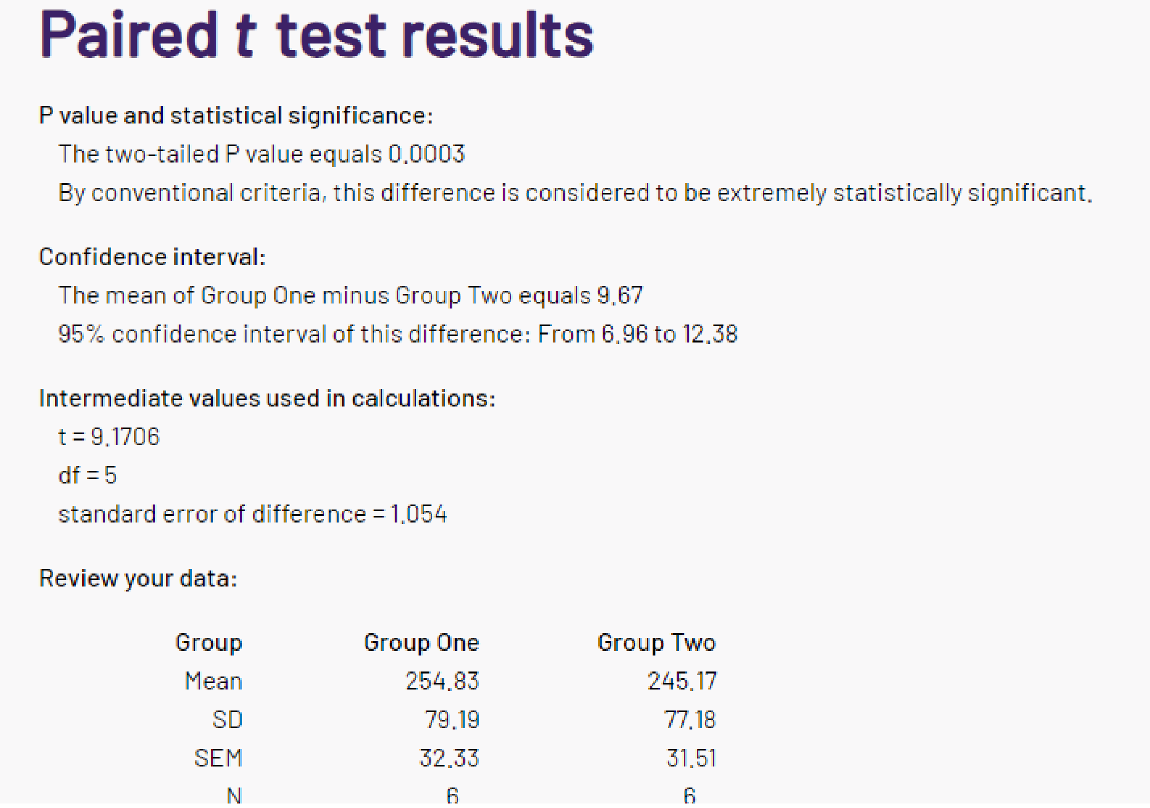
A few assumptions were formulated for this test. The hypothesis proposed that the independent variables, specifically the locations, can be categorized into two correlated groups: one group assessed through manual counting and the other through automated counting. It was hypothesized that the variations between the two counts conform to a normal distribution and that there are no notable anomalies in the differences between manual and automated counts. The following hypotheses were considered:

$$H_0: N_{ai} = N_{mi} \forall_i \tag{1}$$

$$H_A: N_{ai} \neq N_{mi} \forall_i \tag{2}$$

N_{ai} denotes automated total vehicle count at the site i,
N_{mi} denotes the manual total vehicle count at the site i.

A paired t-test was conducted. The disparity between the human and automated counts, known as manual-automated, was assessed using the t-test. The t-test was performed using GraphPad online, the result is illustrated in Figure 4.



Paired *t* test results

P value and statistical significance:

The two-tailed P value equals 0.0003

By conventional criteria, this difference is considered to be extremely statistically significant.

Confidence interval:

The mean of Group One minus Group Two equals 9.67

95% confidence interval of this difference: From 6.96 to 12.38

Intermediate values used in calculations:

t = 9.1706

df = 5

standard error of difference = 1.054

Review your data:

Group	Group One	Group Two
Mean	254.83	245.17
SD	79.19	77.18
SEM	32.33	31.51
N	6	6

Figure 4. Results of paired t-test of the manual and automated vehicle counts.

Significant disparities between human and automatic counts have been identified. The two-tailed paired t-test assumes that the data conforms to a normal distribution. A normality test was utilized to assess if the sample conforms to a normal distribution, considering the restricted sample size of the paired t-test. In this case, the disparities between the human and automatic tallies were investigated. The test indicated that the data about the difference between the automatic and manual counts did not follow a normal distribution. The small size of the data set and the use of random sampling contribute to the absence of a normal distribution. Hence, this could elucidate why the paired t-test fails to ascertain the resemblance between the human and automated counting data.

3.4 Comparison With Other Methods

Table 3 displays the accuracy of several automated counting techniques, including the one used in this research.

In their study, Patrick McGowen and Michael Sanderson found that the precision of the pneumatic road tube counter was around 99 percent [15]. Moreover, the level of inaccuracy in a typical 15-minute traffic count is around 10%, but the average error in a daily traffic count might be almost nil [15]. These data suggest that the errors in both positive and negative counting offset one other, so concealing the extent of inaccuracy. The magnitude of speed and classification mistakes was much greater. These results raise skepticism about the precision of pneumatic road tube counts in measuring traffic volume.

A piezoelectric sensor is activated by the transmission of mechanical energy when a vehicle drives over it. The process of converting mechanical energy into electrical energy is then analyzed for vehicle counts. Based on studies, the piezoelectric sensor can accurately count traffic with a 99 percent level of precision [16].

According to Chen-Fu Liao [17], inductive loops have the potential to provide transportation authorities with more accurate, reliable, and comprehensive traffic performance statistics. To assess the precision of traffic counts, the author conducted many tests on inductive loops [17]. The findings indicated that the level of accuracy was around 99 percent.

Mattias Gustafsson and Sebastian Hjelm [14] developed methods that enable the automated counting of traffic using pre-recorded footage. To enhance the accuracy of automatic traffic counts, high-resolution video was collected. The researchers created and evaluated many neural network models that achieved an accuracy of over 90% in detecting and quantifying automobiles in various situations [14].

Pereira et al. [13] developed a computer software that automates traffic counting. Their focus was primarily on achieving high accuracy and speed in counting. The program was described in detail on pages 210 to 239 of their publication. According to the authors, the algorithm's accuracy is 60% to 70%. The examination determined that the program's incapacity to quantify traffic is attributable to using low-resolution footage.

The algorithms generated in this study exhibit a level of accuracy of 96%, which is higher than earlier computer algorithms. Although piezoelectric sensors and pneumatic road tube counts are more accurate than current methods, they are also more expensive, especially for smaller-scale projects. Thus, considering both cost and accuracy, the suggested algorithms might be a suitable alternative.

Table 3: Comparison of different vehicle detection algorithms

Algorithm	Accuracy
Piezoelectric sensor	99%
Pneumatic road tube counting	99%
Inductive loops	90%
[13] LWR Lighthill–Witham–Richards	60-70 %
[14] Mattias Gustafsson and Sebastian Hjelm	90%
Current study	96%

3.5 Program Limitations

The software's effectiveness is diminished when vehicles pass at high velocities (Figure 5) since their fleeting presence in the footage does not provide enough time for the algorithm to detect them. Occasionally, the system fails to identify these cars precisely. Conversely, the vehicles travel at a diminished velocity and may come to a complete stop while waiting to exit the parking area. In this case, the system notably efficiently detects and measures autos.

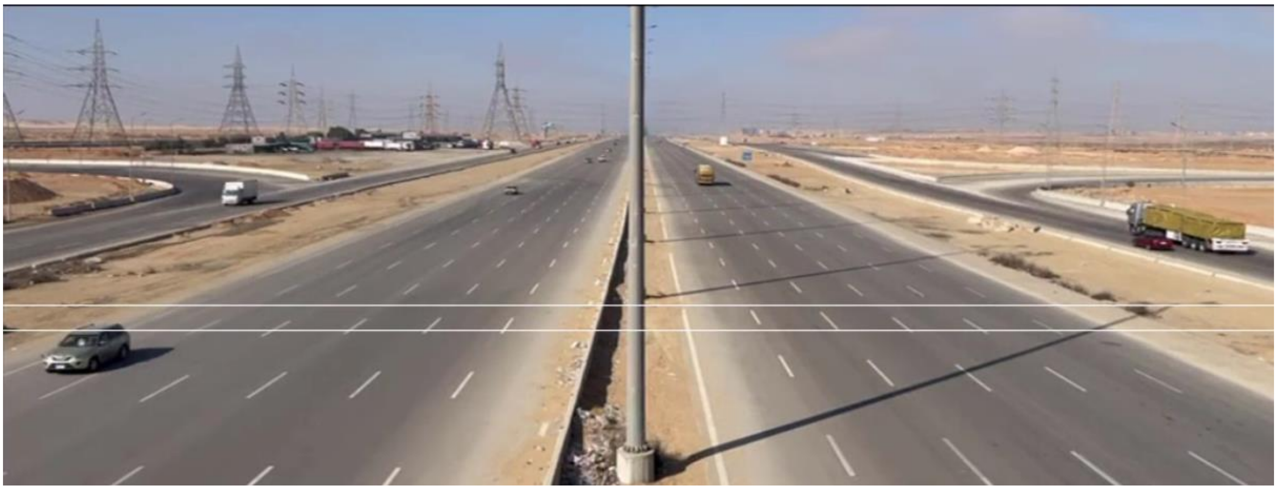


Figure 5. The software's effectiveness is diminished when vehicles pass at high velocities.

The camera angle plays a vital effect in reducing the accuracy of counts. Many cars can be seen when the camera is placed near the entrance, taking up a substantial part of the frame. At times, specific parts of the vehicles may protrude beyond the frame's limits.

Precipitation hinders the camera lenses, resulting in a below-average video recording, and the program cannot accurately tally. Visibility is critical in determining video quality, as less visibility results in inferior video quality and ineffective counting. Reduced visibility can be attributed to causes such as precipitation, decreased illumination, twilight conditions, and gloomy skies.

The algorithm does not detect overlapping vehicles, resulting in an imprecise count. This phenomenon arises when two vehicles simultaneously arrive or depart, causing their images to overlap within the video frame. The program's counting speed depends on the hardware platform. The rate at which counting occurs is directly correlated with the computer's configuration.

3.6 Discussion

The proposed automated vehicle counting model utilizes OpenCV and the YOLO object detection model to determine the number of cars in pre-recorded footage precisely. The effectiveness of the proposed automated counting method was evaluated by statistical analysis, which entailed comparing the automated counts with the manual counts. This test evaluated the global error, error in categorization, and error in intervals of the computerized counting approach. The automated counting method exhibited a precision rate of around 96%, with most inaccuracies arising from underestimating vehicle quantities. When evaluating the precision of alternative methods, it is crucial to emphasize its constant and dependable nature. Based on the automated analysis, it takes around 90 minutes to evaluate 60 minutes of recorded film. The proposed model can be employed in real-time for many applications, such as parking management, traffic safety analysis, intelligent transportation systems (ITSs), traffic management systems, and traffic congestion monitoring. The study indicates that the suggested paradigm offers several significant advantages. These features include accurate traffic counts with a precision rate of 96 percent, the ability to customize time intervals for counting vehicles in specific directions, which is essential for many planners, engineers, and policymakers, and cost-effectiveness, especially for smaller transportation projects.

Nevertheless, the proposed model also displays notable deficiencies. For instance, the program's precision diminishes when the video quality is insufficient, especially in conditions with low illumination, such as during nighttime or under cloudy skies. Furthermore, it faces difficulties maintaining optimal performance when vehicles move at high velocities. Moreover, the model's accuracy may be impaired due to imprecise camera angles and intersecting cars throughout the recording process. To overcome the constraints, it is recommended to utilize high-resolution cameras for capturing movies with enhanced clarity, maintain accurate positioning of the cameras to avoid oscillation caused by wind, and frequently clean the lens before each installation. Furthermore, employing high-resolution computers to save energy and expedite the counting process is recommended.

4. CONCLUSIONS

A Python program using the YOLOv8 and Computer Vision is introduced to count the number and speed of vehicles from a recorded video. The program tested three different roads at different times of the day. The results of the automated counts were compared to the manual counts for the same period. Results show that the computerized counts have an accuracy rate of 96%. Indeed, the program has a high accuracy in vehicle counting, but it still has some difficulties in detecting vehicles precisely in rough weather. However, the achieved accuracy rate at a low cost makes the program a good choice for monitoring highways that extend over long distances where the conventional methods are costly and unbearable. The program is user-friendly, and the user can choose the type of video data, either recorded or streaming, enabling real-time vehicle counting. The program can be modified to add a warning message when a vehicle exceeds the speed limit or the number of vehicles passing through a fixed by minute exceeds a specific amount, enabling traffic congestion prediction.

DECLARATIONS

Author Contributions: Conceptualization, S.K., M.S., A.G.; investigation, S.K., M.S., A.G.; writing— original draft preparation, M.S., A.G.; writing—review and editing, S.K., M.S., A.G.; supervision, S.K., A.G.

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

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] R. Kundu. (2024). Image Processing: Techniques, Types, & Applications. Accessed: July 29, 2024. [Online]. Available: https://www.v7labs.com/blog/image-processing-guide

[2] C., Ridder, O., Munkelt & H. Kirchner. "Adaptive background estimation and foreground detection using kalman-filtering". In Proceedings of international conference on recent advances in mechatronics, Istanbul, Turkey, 1995, pp. 193-199.

[3] A., Boonsongsrikul, & J. Eamsaard, J, "Real-time human motion tracking by tello EDU drone", Sensors, vol. 23, no. 2, 897, 2023.

[4] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), 1999, pp. 246-252.

[5] M., Chauhan, A., Singh, M., Khemka, A., Prateek, & R., Sen, "Embedded CNN based vehicle classification and counting in non-laned road traffic," In Proceedings of the tenth international conference on information and communication technologies and development, 2019, pp. 1-11.

[6] B. Li and R. Chellappa, “A generic approach to simultaneous tracking and verification in video,” IEEE Trans. Image Process. vol. 11, no. 5, pp. 530–544, 2002.

[7] J. S. Liu, R. Chen, and T. Logvinenko, “A Theoretical Framework for Sequential Importance Sampling with Resampling,” in Sequential Monte Carlo Methods in Practice, A. Doucet, N. de Freitas, and N. Gordon, Eds., New York, NY: Springer, 2001, pp. 225–246.

[8] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, “Automatic traffic surveillance system for vehicle tracking and classification,” IEEE Trans. Intell. Transp. Syst., vol. 7, no. 2, pp. 175–187, 2006.

[9] R., Girshick, J. Donahue, T., Darrell, et al., "Rich feature hierarchies for accurate object detection and semantic segmentation", In Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587.

[10] W., Liu, D., Anguelov, D., Erhan, et al., "Ssd: Single shot multibox detector". In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, pp. 21-37.

[11] J., Redmon, S., Divvala, R., Girshick, & A., Farhadi, "You only look once: Unified, real-time object detection," In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016 (pp. 779-788).

[12] T., Sarker & X., Meng, "Traffic Signal Recognition Using End-to-End Deep Learning," In Tran-SET, 2022, (pp. 182-191).

[13] M. Pereira, P. Boyraz Baykas, B. Kulcsár, and A. Lang, “Parameter and density estimation from real-world traffic data: A kinetic compartmental approach,” Transp. Res. Part B Methodol., vol. 155, pp. 210–239, 2022.

[14] S. Hjelm and M. Gustafsson, “Vehicle Counting Using Video Metadata,” LU-CS-EX 2018-13, 2018, Accessed: May 07, 2024. [Online]. Available: https://lup.lub.lu.se/student-papers/record/8962789/file/8962790.pdf

[15] P. McGowen and M. Sanderson, “Accuracy of Pneumatic Road Tube Counters,” presented at the Institute of Transportation Engineers (ITE). Western District Annual Meeting, 2011, May 2011. Accessed: May 07, 2024. [Online]. Available: https://trid.trb.org/View/1257430

[16] P. Bellucci and E. Cipriani, “Data accuracy on automatic traffic counting: the SMART project results,” Eur. Transp. Res. Rev., vol. 2, no. 4, Art. no. 4, Dec. 2010, doi: 10.1007/s12544-010-0039-9.

[17] C.-F. Liao and the University of Minnesota. Department of Mechanical Engineering, “Investigating Inductive Loop Signature Technology for Statewide Vehicle Classification Counts,” MN/RC 2018-31, CTS#2018006, Oct. 2018. Accessed: May 07, 2024. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/64623