**DOI.** 10.1109/ICSTCC59206.2023.10308450

# A Comparative Study of Car Tracking Algorithms for Road Traffic Surveillance Applications

Ana Alexandra Brad
*Department of Computer and Information Technology*
*Politehnica University of Timisoara*
Timisoara, Romania
ana.brad99@gmail.com

Maria Cristina Brad
*Department of Computer and Information Technology*
*Politehnica University of Timisoara*
Timisoara, Romania
maria.brad99@gmail.com

Mihai V. Micea*
*Department of Computer and Information Technology*
*Politehnica University of Timisoara*
Timisoara, Romania
mihai.micea@cs.upt.ro

***Abstract*—In this paper, we approach the important topic of video surveillance systems which automatically identify and track moving vehicles in different environments. A comparative evaluation has been performed on four algorithms which have been adapted for counting vehicles. Among these algorithms, two of them utilize contour detection techniques to accomplish this task, while the remaining two rely on feature detection techniques to count vehicles passing through a designated region of interest using both contour detection and feature detection techniques. The proposed algorithms have been implemented and tested on a set of video sequences and an accuracy measure was computed to determine the most effective algorithm. The results show that the YOLO-based algorithm has the best average accuracy, while the Algorithm 1, which is based on background detection and tracking, has the most consistent performance.**

***Keywords—Traffic surveillance, car detection, Haar Cascades, OpenCV, YOLO***

## I. INTRODUCTION

Video surveillance and tracking systems provide an automatic way to identify people, objects, or events of interest in different types of environments. Typically, these systems consist of dedicated image/ video capturing devices installed in the surveilled locations, along with specialized modules and applications for image sequencing processing, identification and tracking of the movement, objects and/ or people, as well as for data recording and event notification. Identifying moving objects in a video sequence is a fundamental and critical task in video surveillance, traffic monitoring and analysis, human detection and tracking.

This article presents and discusses four algorithms which have been adapted for vehicle detection and counting. Each algorithm employs a unique approach and utilizes a range of techniques, including machine learning, to achieve its objectives. These algorithms can be classified into two categories: two of them involve tracking of the vehicles and displaying the current count near their respective centers, while the other two algorithms count the cars passing through a predefined, designated region of interest. The former category, which focuses on tracking the vehicles, employs contour detection and includes an algorithm that uses YOLO [1]. On the other hand, the second category of algorithms utilizes feature detection over a region of interest, one of them also employing the Haar Cascades method.

However, the issue of calibration occurred due multiple detection of vehicles or background noise. To address it, each algorithm has a configuration file containing calibrations for the respective video. A current constraint is that this process can only be carried out manually, as it depends on the lighting conditions and the presence of background noise.

The four vehicle tracking algorithms have been implemented and tested on a set of video sequences and a comparative evaluation of their performance and accuracy have been made, to determine the most effective algorithm.

## II. STATE OF THE ART

Moving object detection and tracking in video sequences continues to be a topic of major importance and interest in the current scientific and technical literature, with a major impact on a large variety of applications, including road traffic surveillance systems [2], [3].

The authors of [4] present a technique for detecting moving objects by combining Inertial Measurement Unit (IMU) sensors and instance image segmentation. The approach includes using a detector to extract feature points and determining the initial fundamental matrix based on IMU data. The feature points are classified using the epipolar line, and the fundamental matrix is iteratively calculated to minimize classification errors resulting from matching background feature points. The effectiveness of the proposed method was verified using real-world acquired sequences and compared with existing techniques.

In [5], a novel technique for object tracking was introduced, which addresses issues related to changes in scale and occlusion. This method employs the Mean Shift Algorithm [6] to efficiently track the color characteristics of the object. The approach is not only more efficient than other methods, but can also detect target occlusion reliably by using the Bhattacharyya coefficient [7]. By extrapolating an object's motion during occlusion, the technique can accurately estimate its position and provide a foundation for tracking through occlusion.

In the recent years, machine learning based approaches developed rapidly and became intensively studied and used in the vehicle detection and tracking area. For instance, object detection using *Haar* feature-based cascade classifiers is an efficient object detection method, proposed in [8]. This is a machine learning based approach where the Cascade Feature was trained with positive and negative images, and then used to detect objects in other images.

Fast Region-based Convolutional Neural Network (Fast R-CNN) [9], which is considered a benchmark in terms of accuracy, has relatively low speed. To improve the detection efficiency, the You Only Look Once (YOLO) method was proposed in [1]. Compared to the approach adopted by classical object detection algorithms, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes of moving regions and class membership probabilities simultaneously. The YOLO algorithm works by dividing the image into *N* grids, each having a dimensional region equal to *S*x*S*. Each of these *N* grids is responsible for detecting and locating the object contained.

The effectiveness of using Deep Neural Network-based (DNN) background subtraction for detecting moving objects is the focus of [10]. Although previous research has shown DNN-based methods outperform traditional background modeling, the factors contributing to their success are not yet fully comprehended. The authors of this paper examine how a DNN behaves by studying feature maps across all layers and identifying crucial filters that improve detection accuracy. Their analysis reveals that the DNN leverages subtraction operations in convolutional layers and thresholding operations in bias layers, as well as produces filters specific to the scene that help reduce false positives caused by dynamic backgrounds.

The categorization of lanes into Normative-Lanes and Non-Normative-Lanes, and the setting up of corresponding Regions of Interest (ROIs) for vehicle counting is an innovative way of counting vehicles [11]. The ROIs are positioned directly below the camera and set to a length less than the safe vehicle spacing, with the width of the Non-Normative-Lane ROI being the same as the distance between two Normative-Lanes. The authors also explain the use of detection lines and detection areas to improve the accuracy of vehicle counting, and the normalization of ROIs for convenience in research.

## III. Methodology

### A. Algorithm 1

The first algorithm, implemented and studied in this work is based on moving object (vehicle) detection and tracking in video streams. The method starts with a calibration phase in which a previously detected background image is converted to grayscale and then loaded as ground truth. The algorithm then processes the frames one by one, performing moving object detection by frame difference, between the current and the background frames.

The algorithm keeps track of the vehicle positions and assigns a unique identification number to each vehicle. The code examines each vehicle's new position for subsequent frames by comparing it to the prior position. The vehicle is deemed to remain in the same position and its position is updated if the new position is within a predetermined distance threshold [12].

The identification number and the counter of detected vehicles are also displayed on the video stream, while the vehicle positions are marked using circles and text labels.

### B. Algorithm 2

The second algorithm implements a vehicle counting method that starts by computing the difference between the current frame and the background image previously determined. It then applies the Thresholding, Erosion, and Dilation image processing operations to the resulting image in order to enhance the vehicle edges to help establish its bounding box. The bounding boxes that meet the specified criteria were marked with circles, and the vehicles which cross a designated line in the images are counted.

The algorithm displays the marked vehicles and the total number of vehicles detected on the screen. We also added the coordinates of the region of interest to determine the location of the segment used for counting the vehicles.

### C. Algorithm 3

This algorithm was adapted from [12] and it involves detecting objects in the video frame by object recognition. We have applied a parameter that restricts the position of cars below a specified y-coordinate.

For each frame of the video, the algorithm detects any cars in the specified region of interest using the YOLO object recognition method [1]. It determines the coordinates of the center of the cars in each frame and calculates the distance between them to determine if it is the same car:

$$D = \sqrt{\left(C_2(x) - C_1(x)\right)^2 - \left(C_2(y) - C_1(y)\right)^2} \quad (1)$$

where $D$ is the computed distance, and $C_1(x, y)$ and $C_2(x, y)$ are the centers of the cars.

While keeping track of the coordinates of the center of each detected car, the algorithm also assigns it a unique identification number which is shown on the screen next to each vehicle. It then also displays the count of detected cars.

### D. Algorithm 4

The fourth algorithm uses the Haar Cascade Classifier [8] to detect moving vehicles. A set of important parameters are implemented in our adaptation of this algorithm, such as:

- the minimum and maximum width and height of the bounding rectangle,
- the coordinates of the crossing line,
- the allowed offset value for the line.

The pre-trained classifier was applied to each frame in order to detect the cars and a bounding box was drawn for each recognition.

The process then calculates the coordinates of the center of detected cars, stores them in a list and checks if the center is within the allowed offset range of the crossing line. If it is the case, a counter is incremented, and the coordinates are removed from the list. The counter keeps track of the total number of cars that have passed through the line.

Finally, the algorithm displays the current frame with the detected cars and the current counter value.

## IV. Results

The four car tracking algorithms have been implemented in Python language, using the PyCharm environment, developed by JetBrains [13]. The processing architecture is based on an Intel Core i5-4590 CPU with a frequency of 3.30GHz, 8.00GB RAM and a 64-bit Operating System.

Three distinct test sets have been used to evaluate the car tracking algorithms. The first video sequence [14] contains vehicles driving on a highway and passing under a bridge, on which the camera is located. The frames have been captured by a stationary camera, during daytime, with high-contrast.

The second test sequence [15] contains a two-way highway surrounded by a green field and a forest. 20 seconds were used of out this footage. For simplicity reasons, only the left lane was taken into account in this sequence in all of the three cases. The footage was captured during daytime and can be described as stationary.

TS3, the third video sequence [16], has been acquired on a Los Angeles highway during dawn. To avoid the errors in the testing process due to the movement of the video camera at the end of the sequence, that particular part was cut out and only the first 13 seconds were used.

For each of the four considered algorithms, the squares surrounding the vehicles were adjusted in each of the specific files read at the beginning of each code sequence. Therefore, the algorithm was adapted to detect vehicles of the particular dimensions of each specific case.

The algorithms, while powerful and versatile, do have certain limitations. One such example is their limited applicability to trucks. Trucks pose unique challenges due to their size and the algorithms have not been fine-tuned on truck-related data.

Another limitation of the algorithms is their inability to handle videos with significant shakiness or instability. As a result, during the testing phases, the algorithms were predominantly assessed using segments of videos that demonstrated stability. The environment surrounding the objects of interest also poses challenges for the algorithms. For instance, in videos where there is grass or other moving elements, the algorithms might mistakenly interpret the motion of the grass as movement of cars. To address this issue, specific instructions are included in the configuration files guiding the algorithm to focus on specific regions of the video when searching for cars. This approach aims to enhance the accuracy and reliability of the algorithms' object detection capabilities by reducing false positives associated with environmental motion.

The need to specify the maximum and minimum size of vehicles in the configuration files for each video is another issue of the studied algorithms. This requirement arises because these algorithms rely on predefined size constraints to identify and classify objects as vehicles accurately. To make object detection algorithms more flexible and efficient in real-world circumstances where vehicle sizes might change greatly, it can be a continuous area of study and development to overcome the limitations of static size constraints.

A further limitation associated with the algorithms is the reliance on a configuration file for each video, specifically requiring the inclusion of coordinates for the region of interest and an offset value of said region, in the case of Algorithm 2 and 4. The need for manual configuration may be time-consuming and resource-intensive to create or update configuration files for numerous videos, especially in situations where a large number of videos need to be processed. Additionally, relying on manual inputs for the region of interest coordinates and offset introduces the possibility of human error or inconsistencies. Variations in how individuals define and input these values could impact the algorithm's performance and introduce inaccuracies.

The total number of detected vehicles was counted. These values have been compared with the expert observed values, defining the number of vehicles present in the sequence, the number of undetected vehicles, and the number of duplicate cars. The accuracy was computed as follows:

$$A = \frac{V_T - V_M - V_E}{V_T} \cdot 100 \ \ [\%] \qquad (2)$$

where $V_T$ is the number of existing vehicles (ground truth), $V_M$ is the number of undetected (missed) vehicles and $V_E$ is the number or erroneous or duplicate detections.

The evaluation results are presented in the figures and tables below, as follows: Fig. 1 to 3 and Table I for the first car tracking algorithm, Fig. 4 and Table II for Algorithm 2, Fig. 5 and Table III for the third algorithm, and Fig. 6 and Table IV for Algorithm 4, respectively.

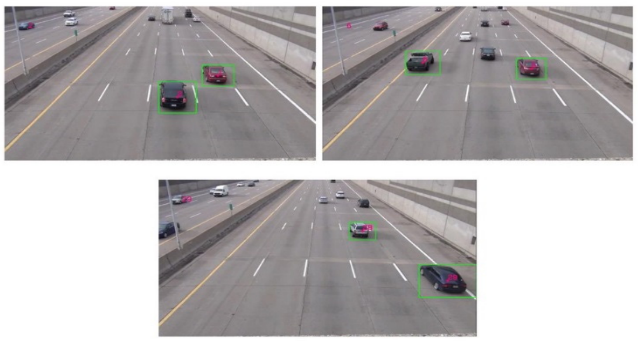

Fig. 1. Graphical results for Algorithm 1 on the first test sequence

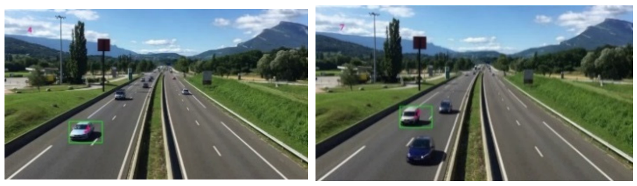

Fig. 2. Display results for Algorithm 1 on the second test sequence (TS2)

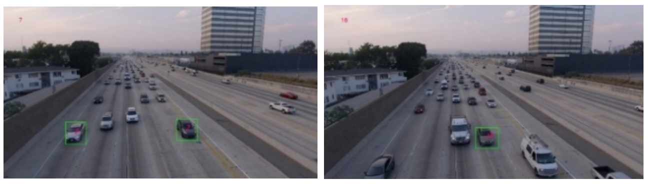

Fig. 3. Display results for Algorithm 1 on TS3

TABLE I. ACCURACY EVALUATION OF ALGORITHM 1

| Metrics | Test Sequences | | |
|---|---|---|---|
| | TS1 | TS2 | TS3 |
| Number of cars ($V_T$) | 53 | 11 | 28 |
| Number of detected cars | 50 | 9 | 30 |
| Number of undetected cars ($V_M$) | 4 | 2 | 2 |
| Number of duplicate cars ($V_E$) | 1 | 0 | 4 |
| Accuracy ($A$) | **90.57%** | **81.82%** | **78.57%** |

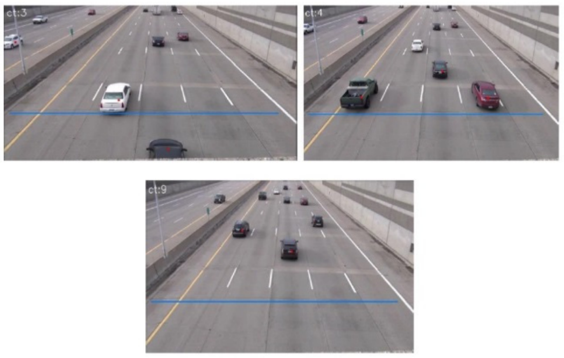

Fig. 4. Display results for Algorithm 2 on TS1

TABLE II. ACCURACY EVALUATION OF THE SECOND ALGORITHM

| Metrics | Test Sequences | | |
|---|---|---|---|
| | TS1 | TS2 | TS3 |
| Number of cars ($V_T$) | 53 | 11 | 28 |
| Number of detected cars | 47 | 12 | 23 |
| Number of undetected cars ($V_M$) | 6 | 1 | 6 |
| Number of duplicate cars ($V_E$) | 0 | 2 | 1 |
| Accuracy ($A$) | **88.68%** | **72.72%** | **75.00%** |

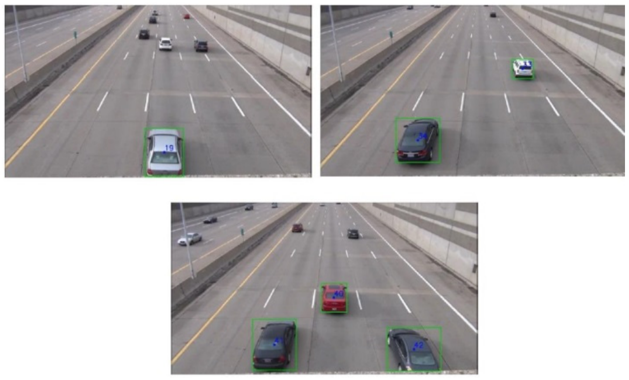

Fig. 5. Display results for Algorithm 3 on the first test sequence

TABLE III. ACCURACY RESULTS FOR ALGORITHM 3

| Metrics | Test Sequences | | |
|---|---|---|---|
| | TS1 | TS2 | TS3 |
| Number of cars ($V_T$) | 53 | 11 | 28 |
| Number of detected cars | 55 | 12 | 36 |
| Number of undetected cars ($V_M$) | 0 | 0 | 0 |
| Number of duplicate cars ($V_E$) | 2 | 1 | 8 |
| Accuracy ($A$) | **96.23%** | **90.91%** | **71.43%** |

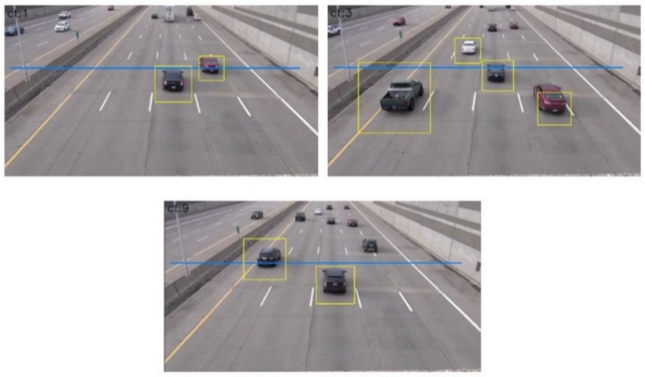

Fig. 6. Display results for Algorithm 4 on TS1

TABLE IV. ACCURACY EVALUATION OF THE FORTH ALGORITHM

| Metrics | Test Sequences | | |
|---|---|---|---|
| | TS1 | TS2 | TS3 |
| Number of cars ($V_T$) | 53 | 11 | 28 |
| Number of detected cars | 46 | 10 | 27 |
| Number of undetected cars ($V_M$) | 14 | 3 | 3 |
| Number of duplicate cars ($V_E$) | 7 | 2 | 7 |
| Accuracy ($A$) | **60.38%** | **54.55%** | **64.28%** |

For instance, in the case of test sequence TS1 on Algorithm1, the outcomes are highly favorable, with only 4 out of 53 vehicles going undetected, and only one car being double-counted. The resulting accuracy is 90.57%.

The first sequence overall has the highest percentages when it comes to accuracy, as the camera hardly moved at all and the contrast is good. There are no trees on the edges, which have a great influence on determining the background. This is why the cars are very well delimited from the rest of the background.

On the second test sequence, the accuracy obtained was over 72% for the first three algorithms and almost 55% for the last one. In this case, it is the oblique sun light that causes the detection rate to drop. There are cases where two cars overpass each other while some of the tested algorithms detect them as a single one, thus missing several cars. The accuracy of the first algorithm was 81.82%, less than in case of the first test sequence. For Algorithm 2, the accuracy was only 72.72%; one car has not been detected and two cars were detected as duplicated. The third algorithm detects the vehicles with an accuracy of 90.91%. The Haar Cascade algorithm had an accuracy of 54.55%.

The third test sequence is the most unstable in terms of camera position. This is one of the reasons why not all vehicles were correctly detected. The first algorithm achieves an accuracy of 78.57%, the second 75%, the third 71.43% and the fourth 64.28%.

A synthesis of the accuracy comparison of the four studied algorithms is presented in Table V.

TABLE V. COMPARISON OF ACCURACY EVALUATION OF THE CONSIDERED ALGORITHMS

| **Test Sequence** | **Car Tracking Algorithm** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **TS1** | **90.57%** | **88.68%** | **96.23%** | 60.38% |
| **TS2** | 81.82% | 72.72% | 90.91% | 54.55% |
| **TS3** | 78.57% | 75.00% | 71.43% | **64.28%** |
| **Average Accuracy** | **83.65%** | 78.80% | **86.19%** | 59.74% |

## V. CONCLUSIONS

In this paper, we have tested and evaluated four vehicle detection algorithms, two of them performing object tracking and the other two counting the cars which move through an area of interest. The first algorithm processes a region extracted with contour detection techniques, whereas the third one tracks shapes recognized by the YOLO machine learning algorithm. Both of these algorithms track moving objects. The other two algorithms use contour detection for the moving region or the recognition of known shapes using Haar Cascades.

TABLE V presents the results of the accuracy comparison of the four algorithms. The average accuracy for the first algorithm is 83.65%. This algorithm achieves the most consistent accuracy in all the three test sequences (over 78%). The average for the second algorithm in terms of accuracy is 78.8%, almost 5 percent lower than the first.

The YOLO-based algorithm achieves over 90% accuracy in the first two cases and, marginally a better average overall accuracy than Algorithm 1, but drops significantly in the case of TS3 test video sequence, due to the camera movement, which complicates object tracking. The fourth algorithm has the lowest accuracy, at an average of 59.74%.

## References

[1] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 779-788, 2016.

[2] A. Crouzil, L. Khoudour, P. Valiere, D.N. Truong Cong, "Automatic Vehicle Counting System for Traffic Monitoring", Journal of Electronic Imaging, 25 (5), 1-12, 2016.

[3] C. Stauffer, W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking", in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 246-252, 1999.

[4] S. Jung, Y. Cho, K. Lee, M. Chang, "Moving Object Detection with Single Moving Camera and IMU Sensor using Mask R-CNN Instance Image Segmentation", *International Journal of Precision Engineering and Manufacturing*, 22 (6), 1049-1059, 2021.

[5] A. Dulai, T. Stathak, "Mean shift tracking through scale and occlusion", *IET signal processing*, 6 (5), 534-540, 2012.

[6] D. Demirovic, "An implementation of the mean shift algorithm", *Image Processing On Line*, 9, 251-268, 2019.

[7] T. Guillerme, N. Cooper, "Effects of missing data on topological inference using a Total Evidence approach", *Molecular Phylogenetics and Evolution*, 94, 146-158, 2016.

[8] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", in *Proc. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, 2001.

[9] R. Girshick, "Fast R-CNN", in *Proceedings of the IEEE International Conference on Computer Vision, ICCV*, 1440-1448, 2015.

[10] T. Minematsu, A. Shimada, H. Uchiyama, R. Taniguchi, "Analytics of Deep Neural Network-Based Background Subtraction", *Journal of Imaging*, 4 (6), 78, 2018.

[11] S. Kumari, D. Agrawal, "Video Based Vehicle Detection and Tracking using Image Processing", *International Journal of Research Publication and Reviews,* 3 (8), 735-742, 2022.

[12] S. Canu, "Object tracking from scratch - OpenCV and Python", *pysource*, 2021, [Online: https://pysource.com/2021/10/05/object-tracking-from-scratch-opencv-and-python/, Last accessed: Jan. 2023].

[13] * * *, "PyCharm Reference", JetBrains s.r.o., 2021, [Online: https://www.jetbrains.com/help/pycharm/ui-reference.html, Last accessed: Jan. 2023].

[14] * * *, https://github.com/MicrocontrollersAndMore/OpenCV_3_Car_Counting_Cpp/blob/master/CarsDrivingUnderBridge.mp4, [Last accessed: Jan. 2023].

[15] * * *, https://pixabay.com/videos/los-angeles-traffic-california-road-53125/, [Last accessed: Jan. 2023].

[16] * * *, https://github.com/ahmetozlu/vehicle_counting/blob/master/src/HSCC%20Interstate%20Highway%20Surveillance%20System%20-%20TEST%20VIDEO.mp4, [Last accessed: Jan. 2023].