Automatic vehicle counting area creation based on vehicle Deep Learning detection and DBSCAN

Gerardo Alvarez Piña
Maestría en Ciencias computacionales
Universidad Autónoma de Guadalajara
Guadalajara, Jalisco, Mexico
jesusg.alvarez@edu.uag.mx

E. Ulises Moya-Sánchez
Coordinación general de innovación and Universidad Autónoma de Guadalajara
Gobierno de Jalisco
Guadalajara, Jalisco, Mexico
eduardo.moya@jalisco.gob.mx

Abraham Sánchez-Pérez
Coordinación general de innovación
Gobierno de Jalisco
abraham.sanchez@jalisco.gob.mx

Ulises Cortés
HPAI
Barcelona Supercomputing Center/UPC
Barcelona, Spain
ulises.cortes@bsc.es

***Abstract*—Deep learning and high-performance computing have augmented and speed-up the scope of video-based vehicles' massive counting. The automatic vehicle counts result from the detection and tracking of the vehicles in certain areas or Regions of Interest (ROI). In this paper, we propose a technique to create a counting area with different traffic-flow directions based on YOLO and DBSCAN You Only Look Once version five (YOLOv5) and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). We compare the performance of the method against the manually counted ground truth. The proposed method showed that it is possible to generate the ROIs (counting areas) according to the traffic flow using deep learning techniques with relatively good accuracy (less than 5 % error). These results are promising but we need to explore the limits of this method with more street-view configurations, time and other detection and tracking algorithms, and in an HPC environment.**

***Index Terms*—Deep learning, vehicle counting, DBSCAN**

I. Introduction

The count of on-road vehicles can have many applications such as traffic management, signal control, urban planning, and most recently the evaluation of citizen mobility due to the COVID restrictions [1], [2]. Automatic detection and counting of vehicles in a video is a challenging task and the machine vision vehicle counting approach is an integrated procedure comprised of detection, tracking, and trajectory processing [3].

The use of video cameras presents a non-intrusive approach to obtaining vehicle counts. However, computer vision algorithms are an expensive computational method. This study is motivated by the need to present an automatic vision-based counting system that addresses the challenging real-world vehicle counting problem. In our experience, analyzing more than 30 CCTV cameras and around 2000 hours of traffic flow videos the visual understanding of the scene to define the counting area was a very time-consuming task. To define this area it is necessary to take into account the traffic direction, the projected geometry, and the region of interest. Moreover, a compromise has to be chosen on its size i.e. the zone has to be large enough to avoid too many false positives and small enough to count every vehicle whatever its size.

In this context, we propose a new method to generate a counting zone based on the You Only Look Once version five (YOLOv5) [4] detection and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering [5]. For this work, we present the results using a GPU implementation because it was five times faster than our CPU implementation. We conducted experiments on three public videos, and the proposed method showed good performance in terms of error comparison with humans. In summary, this study provides the following contributions:

- A new vehicle-counting area strategy is presented. The developed strategy exploits the traffic flow density information to obtain an automatic counting area.

This method could help to automate the counting of vehicles in a fully integrated and automated pipeline.

II. Background

YOLO [4] is one of state of the art real-time[1] detector. Their architecture is divided into three main parts: i) model backbone mainly used to extract important features from the given input image, ii) model neck is mainly used to generate feature pyramids to generalize well on object scaling, and iii) model head mainly used to perform the final output vectors with class probabilities, object scores, and bounding boxes. YOLO performance is a trade-off between the size (number of convolutional layers) and the number FLOPS. The reported performance of small YOLOv5s (in Pytorch) is 17.4 FLOPS processing time frame of 4.2 ms (inference, in V100 GPU). In this work, we present the results on the small version (YOLOv5s) with 213 layers, and 7,225,885 trainable parameters.

[1]Using a GPU

 DOI 10.1109/CLUSTER51413.2022.00069

The other machine learning tool that we use was the DBSCAN [5] for clustering the points. This method is a point-density clustering algorithm. The basic idea behind is that given a set of points, it is possible to group together if many points are nearby neighbors. It is important to note that DBSCAN has a notion of noise, helping to mark the low-density region as outliers points. In this case, high-density regions in combination with geometric priors were enough to create the counting areas. As a non-supervised method, the evaluation of its performance was done by a human.

III. Data

Although we use more than 2000 hours of CCTV from Guadalajara city, in this work we only present the results using public data, see Table I. These videos represent the most common and similar camera/street-view configuration compared with the local (Guadalajara) data. All the videos have the following: 15 frames per second (fps), two-way avenues, simultaneous vehicles, and front perspective. Figure 1 shows one frame of each public video (link video 1, link video 2 link video 3).

TABLE I
Characteristics of the public videos.

Video ID	Max Time	Mean vehicles per min	Lane per way
1	34.08 min	150	3
2	118.45 min	50	3
3	14 min	75	2

IV. Methods

The proposed method is divided into two main stages. First, detects and tracks the vehicles in the videos, second, selects a high-density area and then conducts the clustering using the vehicle centroids. Figure 3 introduces a simplified illustration of the main steps in our method.

A. Detection and tracking

As a first step, we detect the vehicles in all field of view of the video frame. The vehicle detection was performed using a pre-trained deep learning model, YOLOv5 [4] with PyTorch. During this process, the time (frame number) and geometric information (centroid and box size) are recorded. Next, we use the time and geometric information to make the tracking. To put it simply, during this process we take into account the orientation and the distances from the centroid to the nearest vehicles between frames.

B. Counting area creation

In this stage, a selection and clustering process is done. The selection of centroids of interest is based on the detection-tracking outcomes (centroid, ID, box size, frame, among others) and some geometric priors (see Figure 3). We assume that the view of the camera has a projective perspective. As a result, we select the apparently bigger boxes (closer to the cameras). One of the main reasons to do it is because the tracking performance decreases significantly in the remote areas of the field of view. The clustering process is based on the DBSCAN clustering algorithm [5]. We choose this clustering method after comparing the results with K-means, and Gaussian mixture methods. Finally, with the DBSCAN centroids and detected-box limits we define a limit of the entry/creation zone (red polygon) and counting/exit zone (green polygon).

Fig. 1. One example frame of each video used in this work. From top to down: video 1, video 2, video 3.

It is important to note that the proposed method depends on traffic-flow density, traffic direction, time, and geometry of the field of view. These limitations are discussed in the results and conclusion sections.

V. Results and Analysis

An example of the proposed counting area and the human-proposed counting area is presented in Figure 4. It is possible to see that the proposed method is capable of generating two counting areas for each traffic direction. Moreover, the counting areas are bigger to facilitate the count of buses and trucks.

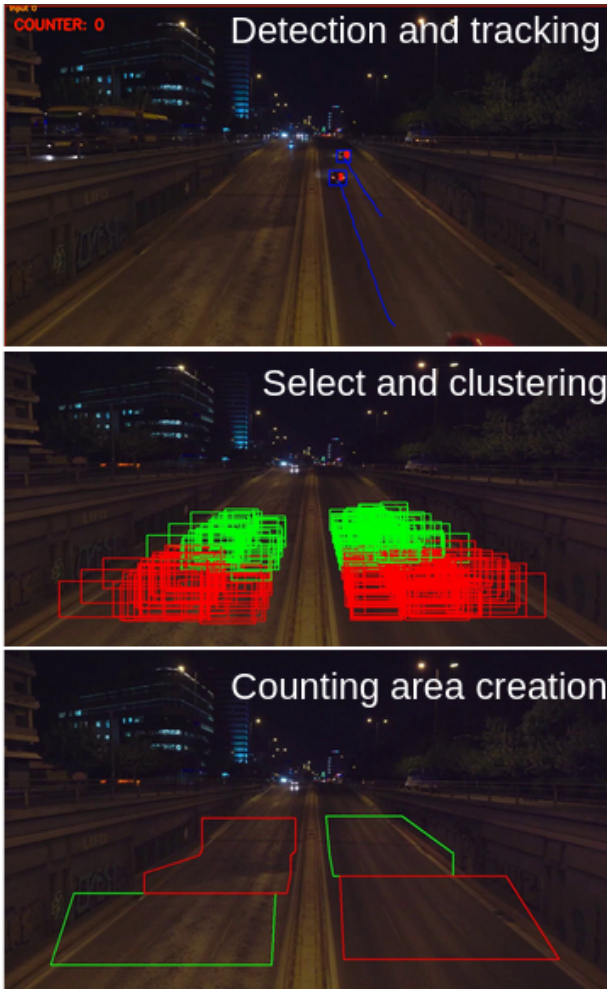

Fig. 2. Counting area creation workflow. The detection and tracking is represented by a blue box and blue line. The section and clustering is based on the boxes (green and red) represented one vehicle ID. The counting areas are represented by a red and green polygons (entry and exit zones).

In Table II we present the numerical results of each video. The vehicle counting error in all the cases is less than 5 %. In addition, this error is reduced if the video duration (and the number of vehicles) to create the counting areas increases. The ground truth was obtained by human counting. According to our most recent experiments counting areas with less than 50 vehicles are not well defined. It is necessary to explore more in detail where the error occurs, and detect possible bias for the type of vehicle, color, or other features.

TABLE II
COUNTING COMPARISON

ID	time	Left	Right	Total	Ground Truth	Error
1	1 min	75	83	158	151	4.63%
1	2 min	130	169	299	292	2.39%
2	1 min	25	39	64	62	3.23%
2	2 min	37	52	89	88	1.12%
3	1 min	36	31	67	69	2.98%
3	1.5 min	65	55	120	119	0.83%

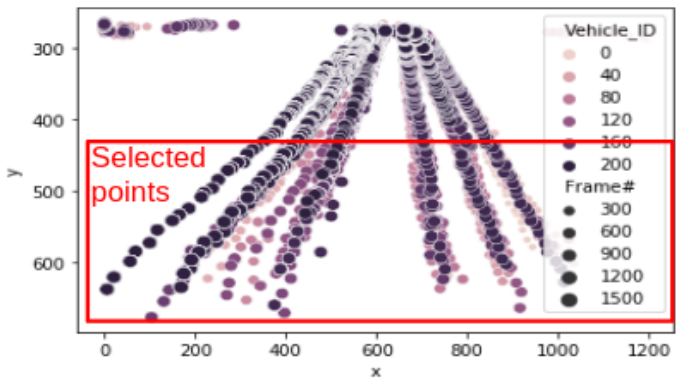

Fig. 3. Selected points of the video based on the geometric priors and box size.

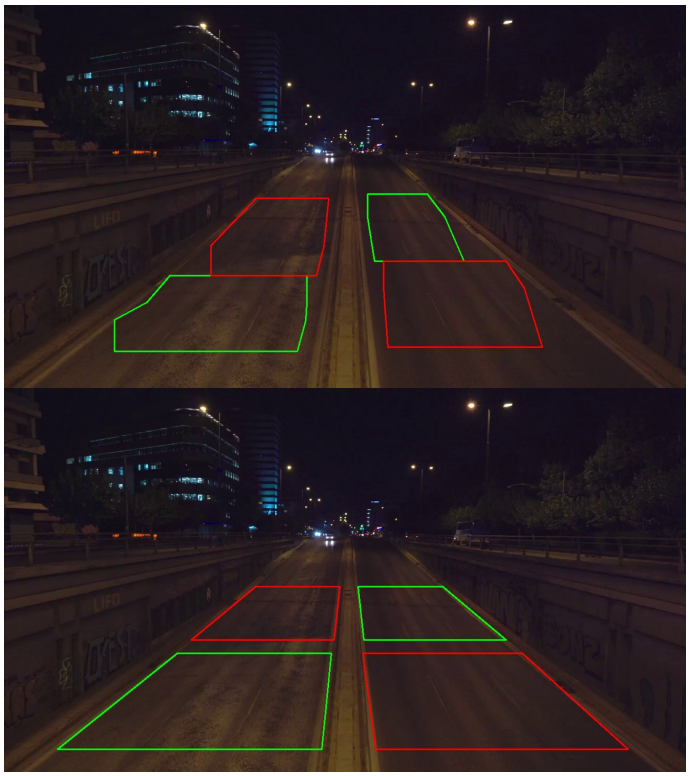

Fig. 4. Example of the automatic (up) counting area created with the proposed method and example of the manual counting area.

VI. CONCLUSIONS

In this work, we show a new method to create counting areas with different traffic-flow directions of on-road vehicles. The presented results show, in all cases errors, below 5%. It is also shown that an increase in video duration reduces the error in the three videos. Although, these results are promising we need to explore the limits of this method in different street-view configurations, video duration, and other detection and tracking algorithms. We would like to test this method in an HPC environment to speed up automatic vehicle counting.

ACKNOWLEDGMENT

We want to thank the CADS-UDeG for providing computing time in the Leo Atrox supercomputer to conduct the inferences of the videos.

REFERENCES

[1] V. Mandal and Y. Adu-Gyamfi, "Object detection and tracking algorithms for vehicle counting: a comparative analysis," *Journal of big data analytics in transportation*, vol. 2, no. 3, pp. 251–261, 2020.

[2] J.-P. Lin and M.-T. Sun, "A yolo-based traffic counting system," in *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 82–85, IEEE, 2018.

[3] A. Gomaa, T. Minematsu, M. M. Abdelwahab, M. Abo-Zahhad, and R.-i. Taniguchi, "Faster cnn-based vehicle detection and counting strategy for fixed camera scenes," *Multimedia Tools and Applications*, pp. 1–29, 2022.

[4] G. Jocher, K. Nishimura, T. Mineeva, and R. Vilariño, "yolov5," *Code repository*, 2020.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.