A Practical Approach for Vehicle Speed Estimation in Smart Cities

Silvio Barra[1], Salvatore Carta[2], Antonello Meloni[2], Alessandro Sebastian Podda[2], and Diego Reforgiato Recupero[2]

[1] Department of Electric Engineering and Information Technology, University of Naples Federico II silvio.barra@unina.it

[2] Department of Mathematics and Computer Science, University of Cagliari {salvatore,sebastianpodda,antonello.meloni,diego.reforgiato}@unica.it

Abstract. The last few decades have witnessed the increasing deployment of digital technologies in the urban environment with the goal of creating improved services to citizens especially related to their safety. This motivation, enabled by the widespread evolution of cutting edge technologies within the Artificial Intelligence, Internet of Things, and Computer Vision, has led to the creation of smart cities. One example of services that different cities are trying to provide to their citizens is represented by evolved video surveillance systems that are able to identify perpetrators of unlawful acts of vandalism against public property, or any other kind of illegal behaviour. Following this direction, in this paper, we present an approach that exploits existing video surveillance systems to detect and estimate vehicle speed. The system is currently being used by a municipality of Sardinia, an Italian region. An existing system leveraging Convolutional Neural Networks has been employed to tackle object detection and tracking tasks. An extensive experimental evaluation has been carried out on the Brno dataset and against state-of-the-art competitors showing excellent results of our approach in terms of flexibility and speed detection accuracy.

Keywords: Surveillance Systems · Convolutional Neural Networks · Object Detection · Smart Cities

1 Introduction

The last few decades have witnessed the increasing deployment of digital technologies in the urban environment in order to improve services to citizens and ensure ever-greater security needs. Such a scenario, combined with the strong scientific and industrial expansion of disciplines such as Artificial Intelligence, Internet-of-Things, and Computer Vision, has led to the emergence of *smart cities* [10]. One of the most significant aspects of this evolution is the increasingly widespread adoption of video surveillance systems and cameras, whose main utility is considered to be that of acting as a deterrent and at the same

time facilitating the identification of perpetrators of unlawful acts or vandalism against public property [21,14,5]. More recently, these infrastructures - integrated into more sophisticated software platforms - have gradually taken on more pervasive roles, from traffic monitoring and automatic access management, to crowd detection, to name a few [16,3,2]. In this context, this work aims to exploit existing city video surveillance systems to provide, in a simple and low-cost way, a solution based on Deep Learning techniques for the automatic detection and estimation of vehicle speed. Contrary to most of the scientific works in the literature, which address the problem through hand-crafted image processing methodologies, the proposed approach innovatively leverages object detection and tracking methods (based on *YOLOv3* and *Darknet*), with an extremely limited setup overhead and a fast *one-off* calibration phase. The system has been adopted by a municipality of Sardinia, an Italian region: more in detail, a couple of cameras in the main road roundabouts have been installed and continuously send videos to the main system installed within the police department. In light of the above, the main contributions of this paper are the following:

- an innovative approach for estimating the speed of vehicles to be applied to fixed-frame video surveillance cameras, which relies on object recognition and tracking methods based on Convolutional Neural Networks, and a detection dynamics built on low complexity heuristics;
- the design of an installation and calibration phase of the proposed method, for the application on new cameras, with reduced execution time;
- an extensive experimental validation of the proposed approach through the public *Brno dataset* and with an in-depth comparison against literature competitors, showing a significant performance of our method both in terms of flexibility and estimation accuracy.

The remainder of this paper is organized as follows. Section 2 explores the related works. Section 3 illustrates the proposed method. Section 4 presents the results obtained from the validation of our approach and comparisons against competitors. Finally, Section 5 ends the paper.

2 Related Works

In recent years, the speed estimation topic has been faced from several points of view, whether the speed is measured from the perspective of other vehicles, from the inside of the vehicle, or the outside (lampposts, traffic lights, and so on). Different sensors may be exploited for the topic and the state of the art offers different solutions for speed estimation, by involving several data sources: as an example, in [12], the authors have exploited a single multi-function magnetic sensor for estimating vehicle speed using three different methods, Vehicle Length based (VLB), Time Difference based (TDB) and Mean Value-based (MVB), so to obtain different reference speed. In [1], the authors have proposed VILDAR (Visible Light Detection and Ranging), built upon sensing visible light

variation of vehicle's headlamp, which has been proven to be able to outperform the accuracy of both RADAR (Radio Detection and Ranging) and LiDAR (Light Detection and Ranging) systems. More information about features, advantages, drawbacks and limitations of RADAR and LiDAR systems is provided in [6]. However, given the multitude of cameras that are currently used for video surveillance purposes [3], most of the approaches are deployed for working on camera streams. Using cameras as the data source for speed estimation is quite complicated, given the fact that some activities are needed before being able to detect the speed of a vehicle: in fact, the car first needs to be detected from the real scenario, then a tracking phase is necessary for understanding its direction and only at the end the speed can be estimated [2]. Also, the camera needs to be calibrated for such a goal [15], according to the context (single/multiple camera/s locations in the scenario) and its/their inner parameters (focal and sensor size). Vehicle speed estimation on camera can be achieved by following two different approaches: the first regards the estimation of the speed of the vehicles in a segment of the road, thus obtaining the average speed of the traffic in that specific section [20,8,11]. In [4] the authors have proposed a full tracking system involving vehicle detection, tracking, and speed estimation based on the displacement in the image of the vehicle concerning the actual position. The second approach is based on the analysis of the n consecutive (or non-consecutive) frames to obtain the pixel displacement of the vehicle in the video [9,24,13]. The operative difference between the two approaches is based on the moment in which the speed estimation module is triggered: while in the first case, the trigger comes from the scenario, i.e. the moment in which a car overcomes the analyzed segment as in [23], in the second case, the trigger is timed from the number of fps the video is taken with.

3 Materials and Methods

Our algorithm is fed with an input video of a road section that may contain a certain number of vehicles (i.e., cars, trucks, motorbikes, and so on). The video is split into frames and for each frame, we perform an object detection task to identify vehicles and follow them during their path. Any recognized object different from a vehicle is discarded. No pre-processing approaches have been applied neither to enhance the image nor for augmenting the local/global contrast of the image. The reason behind this choice is twofold: i) firstly, the method does not need any enhancement of the contrast augmentation approach because the speed detection is an analytical process in which image quality is a side problem; Yolo detection and tracking have not, in fact, been affected at all; ii) secondly, pre-processing stages would need a certain amount of time, which eventually could burden the overall speed detection computation time. Figure 1 shows the diagram of the proposed method. In the next paragraphs, we will give more details on our vehicle speed estimation system.

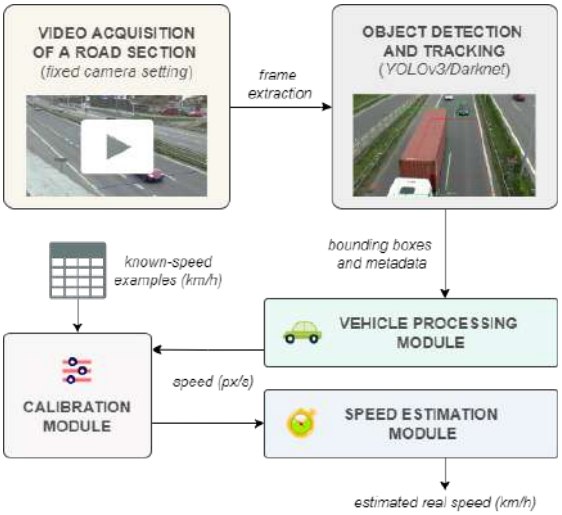

Fig. 1. From the video frames of a road section to the speed in km/h: frames are processed with $Yolo_{V3}$+*Darknet*, which returns the bounding boxes of the detected vehicles. They are used to follow the trajectory of vehicles and estimate their speed in $pixel/s$. Existing real speeds of some vehicles are used to calibrate the system. The *Speed Estimation module* applies the calculated transformation function and returns the speed in km/h.

3.1 Vehicle speed estimation system

Our system exploits *YOLO* (You Only Look Once - a real-time object detection system) [17] with *Darknet* [18] (an open-source neural network framework written in C and CUDA)[3] to locate vehicles in each frame. Differently from other object detection systems, like Fast R-CNN [7] and Faster R-CNN [19] which lie in the field of the Region proposal network, the YOLO family of object detectors, as the name suggests, achieves region analysis and object localization within an image or frame in just one stage. Indeed, the canonical RPN approaches divide the region proposal stage and the object detection stage in two different steps. During a detection session, the video frames are analyzed one by one. If a new vehicle is detected, it will be given an ID and its position will be tracked over the next few frames until it leaves the scene. A small amount of time passes between each frame (usually less than 50 milliseconds) so the position of the same vehicle in two successive frames cannot be too different. If multiple vehicles are identified within the same frame, the approach performs a nearest neighbor computation of the objects within the different frames. In such a case, the tracking method calculates which of the vehicles in the frame is closest to the position of each vehicle in the previous frame. The location found is used to update the one associated with that ID. For example, let us suppose that

[3] https://pjreddie.com/darknet/yolo/

vehicles A and B have been recognized within the frame i. Then, if in frame $i+1$ we identify multiple vehicles, the one corresponding to A will be that for which the distance between its position in frame $i+1$ and the position of the identified vehicle A in frame i is the shortest. One more constraint that vehicles must satisfy is the detection area they must reside within. If they are detected out of the detection area, they are discarded. The detection area is fixed for the entire video sequence and is considered one input of the problem. An example of a detection area is shown in Figure 2, corresponding to the green and red crossed lines. A vehicle is analyzed as soon as it passes the first green light (indicated on top of the figure) and is continuously tracked until it leaves the last green line (that at the bottom of the image). When a vehicle enters the detection area, its entry point and the current frame number are stored. When it exits the area, the distance between the exit point EX and the entry point EN is calculated in pixels and this value is divided by the elapsed time. Time is calculated as the number of frames elapsed between entering and exiting the detection area divided by the number of frames per second of the video. For example, if F_1 is the frame related to the entry point and F_2 the first frame when the vehicle left the detection area, the speed is computed as $\frac{dist(EX,EN)}{\frac{F_2-F_1}{50}}$, assuming the frame rate equal to 50 fps. The speeds returned in $pixels/s$ are converted to km/h by linear interpolation and the real speed of a few reference vehicles known as input. In particular, in the Cartesian plane we construct a graph with the calibration points $X(detectedspeed\ _{(pixel/s)},\ real\ speed\ _{(km/h)})$ (they are known and belong to the input vehicles) and, using the *Least Squares* method, we calculate the equation of the line that best approximates the position of these points. The identified equation (which is computed only once) is used to calculate the speed in km/h from that in $pixels/s$. For example, let us assume we have the following five points (903, 82.93), (910, 79.99), (856, 80.34), (935, 83.70), (810, 68.69), each of type ($pixel/s$, km/h), representing five calibration points known in advance, and the speed of 927 $pixels/s$ of a vehicle just detected. Using the *Least Squares* method we find $m = 0.0862$ and $q = 3.0157$ related to the equation $y = mx + q$. By substituting the x with 927 we obtain an estimated speed of 82.92 km/h, a value relatively close to the real one (82.81 km/h).

3.2 The BrnoCompSpeed dataset

Authors in [22] worked on visual traffic surveillance and captured a new dataset of 18 full-HD videos, each lasted one hour, plus three more videos (of around 10 minutes each) recorded at seven different locations. The videos contain vehicles that are annotated with the exact speed measurements verified with different reference GPS tracks. The dataset is made up of 21 road section videos. These videos are split into six sessions, plus one (manually annotated), called session zero. More in detail, each session includes three videos, one taken from the left camera, one taken from a central camera, and the third one taken from the right camera. We will refer to each video as *Session ID-left|central|right* where *ID* $\in \{0, \ldots, 6\}$ and *left, central,* and *right* depend on whether the video was taken using the left camera, central camera, or right camera. Each session is stored in

a folder that contains the videos of a different road section taken from the three points of view. For each video, there is a file containing vehicle data, a photo of the road section, and an image containing the detection area. In total, in the dataset, there are video recordings and speed data of 20,865 vehicles.

The data files were produced with the Pickle module - Python object serialization - for Python 2.7. The data contained in the Pickle files include:

- **[distanceMeasurement]**: list of measurement points and actual distances.
- **[measurementLines]**: list of the coefficients (a, b, c) of the measurement lines, perpendicular to the road axis, to be used with the canonical form of the line equation $ax+by+c = 0$. A digital image is a grid in which each point (pixel) is identified by a pair of numbers (x, y) and a straight line can be identified by a linear equation such as $ax + by + c = 0$ (canonical equation). To automatically understand if a vehicle is within the detection area and in which lane, a system of linear equations corresponding to the sides of the street and the lanes and the perpendicular lines to the street must be solved.
- **[cars]**: list of data related to running vehicles. Each info includes $laneIndex$ (lane number - 0 on left), $carId$, $timeIntersectionLastShifted$, $valid$, $intersections$, and $speed$. The $intersections$ field is a list containing information about the exact time the vehicle crosses the measurement lines and can be useful to identify it in the video. The value of the $valid$ field determines whether the measurement has to be taken into account or not.
- **[fps]**: frames per second of the video.
- **[invalidLanes]**: list of invalid lanes.
- **[laneDivLines]**: list of the coefficients (a, b, c) of the lines which divide the lanes, to be used with the canonical form of the line equation $ax+by+c = 0$.

In Figure 2, the measurement lines (in red) and the delimitation lines (in green) of the lanes, obtained from the coefficients present in the data file of the Session 0-left of the dataset, are plotted. In Figure 3, one of the vehicles, taken from the Session 0-left video, can be visually identified in the frame corresponding to the exact moment it crosses the measurement line.

Fig. 2. Measurement lines.

Fig. 3. Vehicle identification.

The vehicle data obtained from the videos with five different calibration systems by the authors of the dataset were included in five JSON files per video. Each JSON file includes a list of the identified vehicles with the following items:

- **[id]** -identifier of the vehicle;
- **[frames]** - the frames in which the vehicle appears;
- **[posX]** - X coordinates of the vehicle in each frame in which it appears;
- **[posY]** - Y coordinates of the vehicle in each frame in which it appears.

The BrnoCompSpeed Dataset Evaluation Code is available in the project's GitHub repository[4]. The code in the repository generates the statistics on the errors obtained in the phases of calibration, detection of distances, and estimation of speeds. These statistics include, in addition to the number of the performed measurements, mean, median, 95 percentile, and worst case (absolute and percentage values) for each video of all sessions from 1 to 6. In Table 1 we show the summary of the results obtained by the competitor methods on the Brno dataset. In the next section, we exploit such methods as the baselines for evaluating the performance of the proposed speed measurement system.

Table 1. Competitor's performance on the Brno dataset for each calibration system and for the videos of sessions 1-6. (MAE: Mean Absolute Error; RE: Relative Error)

System	MAE (km/h)	RE (%)
FullACC	8.59	10.89
OptScale	1.71	2.13
OptScaleVP2	15.66	19.83
OptCalib	1.43	1.81
OptCalibVP2	2.43	3.08

4 Experimental Analysis

A statistically significant subset of videos from the Brno dataset has been used to experimentally analyze and validate the proposed speed estimation system. For each video, the vehicles were identified and associated with the real speed measured by the authors of the dataset. The selected piece of video was then analyzed with YOLO + Darknet to identify the positions of the vehicles in each frame. The real speeds of some of the vehicles were used to calculate the transformation parameters from $pixel/s$ to km/h. The purpose of the tests is to compare the results obtained by the authors of the datasets above with those of our speed estimation system. In all the tests, the video analysis - the detection of vehicles and the calculation of their speed, returned in $pixel/s$ - were performed on *Google Colab*. The equation of the line for the linear interpolation, the vehicle speeds in km/h and the statistics were instead obtained using a Python script and offline computing.

[4] https://github.com/JakubSochor/BrnoCompSpeed

4.1 Preliminary Quality Test

We first started the validation of our method by testing the quality of the results obtained for the measurement of the speed using the initial parameters for the detection and calculation of the position of the vehicles.

In this analysis, the center of the bounding box was used to denote the vehicle's position. Then, we estimate the speed for the first 20 vehicles of the video belonging to the *Session 0-left* of the dataset (one vehicle passed on lane 1, fourteen on lane 2, and five on lane 3), by considering the vehicles which transited on lane 2 only. The speed value obtained by the system in $pixels/s$ was multiplied by 3.6 to get $kpixel/h$. Moreover, the first five detected vehicles (see Table 2) were used for the calibration stage. At the last step, the coefficients $m = 0.0158$ and $q = 29.9095$ of the line equation $y = mx + q$ were derived with the least-squares method. The speed in km/h was obtained from the equation by substituting the speed measured in $kpixel/h$ for x. The test produced the following results (data from the first five vehicles, used for calibration, are not considered): standard deviation of 0.0325, mean relative error of 3.27%, mean absolute error of 2.36 km/h.

Table 2. Preliminary quality test: speed estimation results (MAE: Mean Absolute Error; RE: Relative Error.

Vehicle id	**Detected speed (kpixel/h)**	**Detected speed (km/h)**	**Real speed (km/h)**	**MAE (km/h)**	**RE (%)**
0	3252	81,27	82,93	1,67	2,01
3	3276	81,65	79,99	1,66	2,07
4	3084	78,62	80,34	1,73	2,15
5	3367	83,09	83,70	0,61	0,73
8	2967	76,77	74,42	2,35	3,15
9	2864	75,14	73,75	1,40	1,89
10	3305	82,11	81,74	0,36	0,44
11	2838	74,73	70,09	4,65	6,63
12	2916	75,96	68,69	7,27	10,59
14	3258	81,36	81,59	0,22	0,27
16	3339	82,64	82,81	0,17	0,20
17	3093	78,76	75,51	3,25	4,31
18	3100	78,87	77,11	1,75	2,27
19	3087	78,66	76,49	2,17	2,84

We can observe from Table 2 that the errors for vehicles 11 and 12 are much higher than the average. We looked at these vehicles (which we considered outliers) in the video and noticed that they have much bigger dimensions (length and height) than the others in the test. This property affects the size of the bounding box and therefore the position of its center, which is used to define the position of the vehicle itself. The bounding box assumes different sizes at the times of entry and exit from the detection area, introducing errors, especially if the entry or exit lines are close to the edge of the frame.

We also carried out a second test using a different detection point, to see whether the results were confirmed or disproved Indeed, for some vehicles, the centroid of their bounding boxes is not properly computed. As we noticed that its lower right corner is usually more stable and precise, from this test on we used it as the point of reference for the calculation of the position of the vehicle.

Table 3. Results obtained with a different detection point.

Vehicle id	**Detected speed (kpixel/h)**	**Detected speed (km/h)**	**Real speed (km/h)**	**MAE (km/h)**	**RE (%)**
0	3089	83,75	82,93	0,81	0,98
3	2777	79,51	79,99	0,48	0,60
4	2764	79,34	80,34	1,01	1,25
5	2958	81,97	83,70	1,73	2,06
8	2579	76,82	74,42	2,40	3,23
9	2546	76,38	73,75	2,63	3,57
10	2858	80,61	81,74	1,13	1,38
11	2439	74,92	70,09	4,84	6,90
12	2409	74,52	68,69	5,82	8,48
14	2833	80,27	81,59	1,31	1,61
16	2873	80,81	82,81	2,00	2,41
17	2665	77,99	75,51	2,48	3,29
18	2662	77,95	77,11	0,84	1,08
19	2651	77,80	76,49	1,31	1,72

The results of this test are reported in Table 3 and show a standard deviation of 0.0246, a mean relative error of 3.38%, and a mean absolute error of 2.49 km/h (the vehicle data used for calibration are excluded). They also reveal that the variation of the considered detection does not appear to significantly affect, on balance, the accuracy of the method.

4.2 Impact of the Observation Angle

We then investigate the possibility that the angle of observation may impact the goodness of measurement. In particular, with regard to the Brno dataset, the framing is the same for the different lanes considered: this implies a different angle of observation and consequent a different perspective for each lane, and such a configuration could impact the measurement result. To estimate the magnitude of this effect, the video of Session 1 - Central was chosen, with the lower right corner of the bounding box being used as the vehicle location. Consequently, the first 20 vehicles of Session 1 - central (two lanes) - were considered, while the first five vehicles for each lane were used for calibration. Moreover, to automate the calibration and test procedure, specific checks have been implemented to avoid cases of double detection (a bicycle hanging on the back of a camper detected

as an additional vehicle or a vehicle with two bounding boxes almost completely overlapping) or detection in non-interesting areas of the image. The results are shown in Table 4.

Table 4. Impact of the observation angle.

	Lane 1	**Lane 2**	**Average**
Standard deviation	0,0192	0,0053	0.0122
Mean relative error (%)	2.44	1.42	1.93
Mean absolute error (km/h)	1.66	1.17	1.42

We recall that the lane 1 is lateral to the frame, while the lane 2 is central. Hence, from the results in Table 4, it emerges that a greater alignment of the camera with respect to the monitored lane (and therefore a reduced or absent angle of observation), as is the case for central lane 2, also results in less detection error, as intuitively expected.

4.3 Impact of the Detection Area Size

To clarify the possible relationship between the vertical length (i.e., the *height*, with respect to the single frame) of the detection area and the precision of the measurements, we compare the speed errors obtained by considering four different dimensions of the detection area. In this additional case scenario, the lower right corner of the bounding box was used to denote the vehicle's position, and the first 25 vehicles of the Session 1 - central (lane 1) - were considered. Four measurement sessions were thus performed with detection areas of variable height between 200 and 920 pixels (see Figures 4, 5, 6 and 7).

In such a scenario, the results highlight a significant decrease in errors as the size of the detection area increases (see Table 5 and Figure 8).

Table 5. Impact of the detection area size.

Area height (px)	**MAE (km/h)**	**MRE (%)**	**Standard deviation**
200	4.54	6.43	0.0437
400	2.57	2.57	0.0189
620	1.30	1.87	0.0113
920	0.79	1.06	0.0070

4.4 Entry/Exit Point Improved Estimation

At last, we explore a method to improve the precision of the speed estimation by determining the exact entry and exit points in/from the detection area. In this case, the lower right corner of the bounding box was used to locate the vehicle,

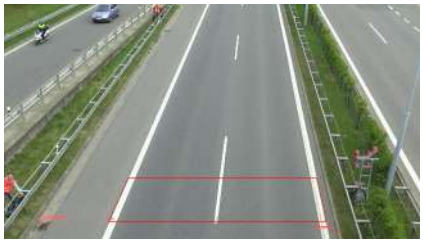

Fig. 4. h = 200 pixels.

Fig. 5. h = 400 pixels.

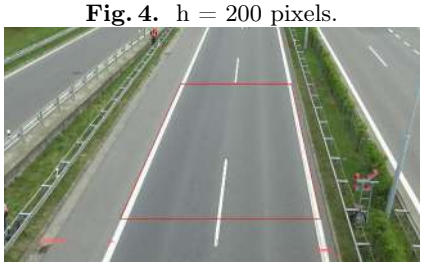

Fig. 6. h = 620 pixels.

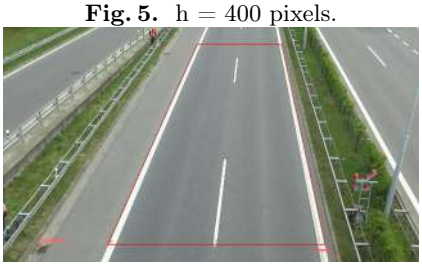

Fig. 7. h = 920 pixels.

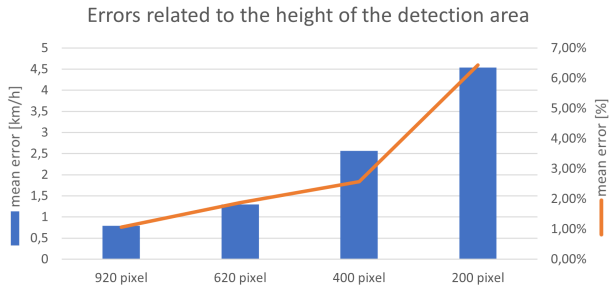

Fig. 8. Errors related to the height of the detection area.

and the first 25 vehicles of the Session 1 - central (lane 1) were considered. Here, the detection area height was fixed at 920 pixels, while the first three vehicles (other than the fastest and the slowest), the fastest vehicle, and the slowest vehicle were used for the calibration.

To implement such an improvement, we just keep the frame just before the vehicle crosses the detection area and the subsequent frame where the vehicle touches it. Similarly, when the vehicle exits from the detection area, we keep the frame where it exits it and the frame just before the exit. The moments when the vehicles enter and leave the detection area can therefore be calculated comparing the vehicle positions in the previous and current frame. If A is the point in the frame right before the vehicle enters the detection area and B is the point in the next frame (where the vehicle touches the detection area), we can compute the line between A and B. As we know the equation of the line of the detection area, we can compute the intersection point I of the two lines

and have the exact coordinate of the entry point. Similarly, we can compute the exact exit point.

The number of frames we take into account to compute the speed in $pixel/s$ is equal to the number of frames elapsed between the frame F_1, when the vehicle enters the detection area, and the frame F_2, the last frame before the vehicle exits the detection area, plus the fraction of the frame right before F_1 and the fraction of frame right after F_2. The fraction of the frame before F_1 is equal to the distance IB divided by the distance AB. With similar calculations we compute the fraction of the frame after F_2.

Table 6. Results obtained by improving the estimation of the entry/exit points.

MAE (km/h)	**MRE (%)**	**Standard deviation**
0.24	0.35	0.0025

In the previous test scenarios, the starting position of the vehicle was considered in the first frame after the vehicle passed the starting point of the detection area and the ending position was considered in the first frame after the vehicle was out of the detection area. Indeed, this led to a minor precision as the vehicle was always some pixels over the starting/ending point of the detection area. The results are shown in Table 6, where such an improvement generates a better performance of the proposed method.

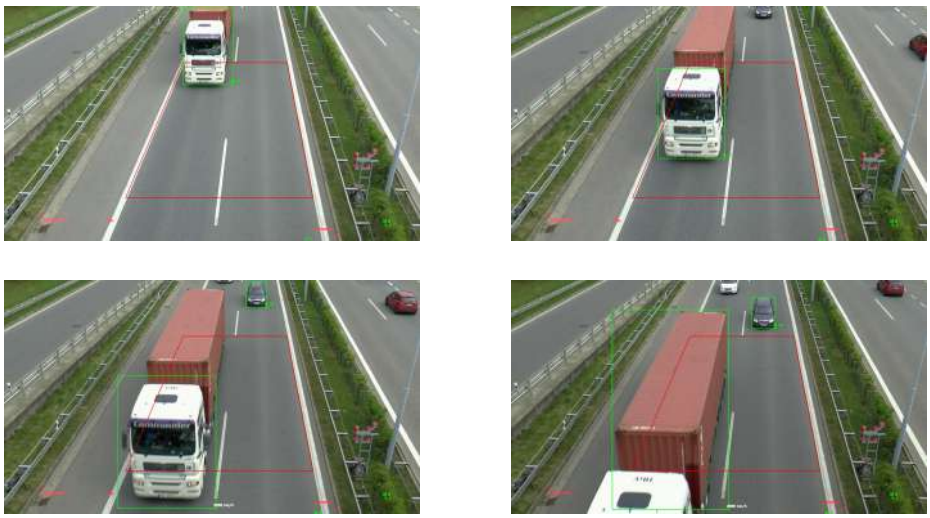

Fig. 9. Inconsistent bounding boxes for the same vehicle.

4.5 Comparison with the state-of-the-art

Finally, we compare our results with those obtained by the authors of the dataset and the other state-of-the-art competitors on the Brno dataset. To do so, we executed a conservative configuration of our algorithm, thus excluding possible improvements identified *ex-post*, which might have suffered from overfitting due to the specific scenario related to the Brno dataset. In addition, we ran the evaluation code available in the Brno project repository for the video related to the Session 1-Central.

Table 7. Summary results of the comparison with the state-of-the-art methods.

	MAE (km/h)	**RE (%)**
FullACC	10,15	13.08
OptScale	3.13	4.12
OptScaleVP2	48.71	62.69
OptCalib	3.12	4.10
OptCalibVP2	7.02	9.06
UniCA *(proposed)*	**1.42**	**1.93**

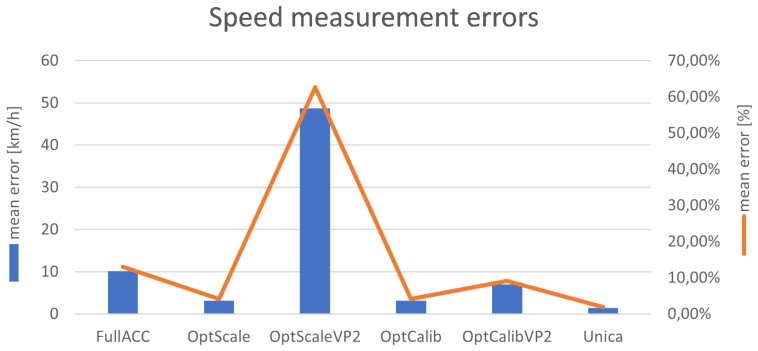

Fig. 10. Summary results of the comparison with the state-of-the-art methods.

The summary data of this comparison are illustrated in Table 7 and Figure 10, where the proposed method is referred to as *UniCA*. The results obtained not only show a clear superiority of the proposed method in terms of lower average error, but are also overall slightly below the 2% (in relative terms) or 2km/h (in absolute terms) thresholds that are used by several world Countries to accredit automatic speed detection systems.

5 Conclusions

The developed algorithm[5], although its operating logic is relatively simple, performs well, with a relatively average error of 1.93% compared to its literature

[5] We publicly release the code at `http://aibd.unica.it/speed_detection.zip`.

competitors, and a low-cost setup and calibration phase that allows for easy extension to existing traffic surveillance systems, provided that an affordable embedded GPU system (e.g., the *NVIDIA Jetson* board) is available to execute the real-time object detection and tracking algorithms.

Through the experiments carried out, we have also highlighted how such results can be further improved through the adoption of specific expedients, which, however, depend on the type of framing and road setting considered. However, some limitations remain. Looking at the video output generated by the system, in some cases, the size of the bounding box changes abruptly in a few frames due to particular occlusion or misdetection conditions that can deceive the YOLO + Darknet framework (see Figure 9 for an example). If this happens in the specific frame in which the distance measurement is performed, higher errors for speed measurement can be generated. A possible way to mitigate this that we will tackle in a forthcoming enhancement of this approach, could be to consider the size of the box in each frame and a moving average of its shape to avoid inconsistent spots changes. We finally point out that the proposed approach - in a revised engineered implementation made more stable and robust - performs well enough to have enabled its usage by the *Monserrato* municipality in Sardinia, Italy, to help the local police identify hazard behaviours of drivers in selected critical road sections.

References

1. Abuella, H., Miramirkhani, F., Ekin, S., Uysal, M., Ahmed, S.: Vildar—visible light sensing-based speed estimation using vehicle headlamps. IEEE Transactions on Vehicular Technology **68**(11), 10406–10417 (2019)
2. Atzori, A., Barra, S., Carta, S., Fenu, G., Podda, A.S.: Heimdall: an ai-based infrastructure for traffic monitoring and anomalies detection. In: 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). pp. 154–159. IEEE (2021)
3. Balia, R., Barra, S., Carta, S., Fenu, G., Podda, A.S., Sansoni, N.: A deep learning solution for integrated traffic control through automatic license plate recognition. In: International Conference on Computational Science and Its Applications. pp. 211–226. Springer (2021)
4. Cheng, G., Guo, Y., Cheng, X., Wang, D., Zhao, J.: Real-time detection of vehicle speed based on video image. In: 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). pp. 313–317 (2020). https://doi.org/10.1109/ICMTMA50254.2020.00076
5. Feldstein, S.: The global expansion of AI surveillance, vol. 17. Carnegie Endowment for International Peace Washington, DC (2019)
6. Fisher, P.: Improving on police radar. IEEE Spectrum **29**(7), 38–43 (1992). https://doi.org/10.1109/6.144510
7. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
8. Gunawan, A.A., Tanjung, D.A., Gunawan, F.E.: Detection of vehicle position and speed using camera calibration and image projection methods. Procedia Computer Science **157**, 255–265 (2019)

9. Kamoji, S., Koshti, D., Dmonte, A., George, S.J., Pereira, C.S.: Image processing based vehicle identification and speed measurement. In: 2020 International Conference on Inventive Computation Technologies (ICICT). pp. 523–527. IEEE (2020)
10. Kunzmann, K.R.: Smart cities: A new paradigm of urban development. Crios **4**(1), 9–20 (2014)
11. Lee, J., Roh, S., Shin, J., Sohn, K.: Image-based learning to measure the space mean speed on a stretch of road without the need to tag images with labels. Sensors **19**(5), 1227 (2019)
12. Li, H., Dong, H., Jia, L., Xu, D., Qin, Y.: Some practical vehicle speed estimation methods by a single traffic magnetic sensor. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 1566–1573 (2011). https://doi.org/10.1109/ITSC.2011.6083076
13. Liu, C., Huynh, D.Q., Sun, Y., Reynolds, M., Atkinson, S.: A vision-based pipeline for vehicle counting, speed estimation, and classification. IEEE transactions on intelligent transportation systems **22**(12), 7547–7560 (2020)
14. Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Uavs for smart cities: Opportunities and challenges. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 267–273. IEEE (2014)
15. Neves, J.C., Moreno, J.C., Barra, S., Proença, H.: Acquiring high-resolution face images in outdoor environments: A master-slave calibration algorithm. In: 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS). pp. 1–8. IEEE (2015)
16. Radu, L.D.: Disruptive technologies in smart cities: a survey on current trends and challenges. Smart Cities **3**(3), 1022–1038 (2020)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
18. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
19. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
20. Schoepflin, T.N., Dailey, D.J.: Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. IEEE Transactions on Intelligent Transportation Systems **4**(2), 90–98 (2003)
21. Shorfuzzaman, M., Hossain, M.S., Alhamid, M.F.: Towards the sustainable development of smart cities through mass video surveillance: A response to the covid-19 pandemic. Sustainable cities and society **64**, 102582 (2021)
22. Sochor, J., Juránek, R., Špaňhel, J., Maršík, L., Široký, A., Herout, A., Zemčík, P.: Comprehensive data set for automatic single camera visual speed measurement. IEEE Transactions on Intelligent Transportation Systems **20**(5), 1633–1643 (2019). https://doi.org/10.1109/TITS.2018.2825609
23. Tourani, A., Shahbahrami, A., Akoushideh, A., Khazaee, S., Suen, C.Y.: Motion-based vehicle speed measurement for intelligent transportation systems. International Journal of Image, Graphics and Signal Processing **10**(4), 42 (2019)
24. Vakili, E., Shoaran, M., Sarmadi, M.R.: Single–camera vehicle speed measurement using the geometry of the imaging system. Multimedia Tools and Applications **79**(27), 19307–19327 (2020)