Khazukov *et al. J Big Data* (2020) 7:84
https://doi.org/10.1186/s40537-020-00358-x

Journal of Big Data

RESEARCH Open Access

Real-time monitoring of traffic parameters

Kirill Khazukov, Vladimir Shepelev*, Tatiana Karpeta, Salavat Shabiev, Ivan Slobodin, Irakli Charbadze and Irina Alferova

*Correspondence:
shepelevvd@susu.ru
South Ural State University, Chelyabinsk 454080, Russia

Abstract

This study deals with the problem of rea-time obtaining quality data on the road traffic parameters based on the static street video surveillance camera data. The existing road traffic monitoring solutions are based on the use of traffic cameras located directly above the carriageways, which allows one to obtain fragmentary data on the speed and movement pattern of vehicles. The purpose of the study is to develop a system of high-quality and complete collection of real-time data, such as traffic flow intensity, driving directions, and average vehicle speed. At the same time, the data is collected within the entire functional area of intersections and adjacent road sections, which fall within the street video surveillance camera angle. Our solution is based on the use of the YOLOv3 neural network architecture and SORT open-source tracker. To train the neural network, we marked 6000 images and performed augmentation, which allowed us to form a dataset of 4.3 million vehicles. The basic performance of YOLO was improved using an additional mask branch and optimizing the shape of anchors. To determine the vehicle speed, we used a method of perspective transformation of coordinates from the original image to geographical coordinates. Testing of the system at night and in the daytime at six intersections showed the absolute percentage accuracy of vehicle counting, of no less than 92%. The error in determining the vehicle speed by the projection method, taking into account the camera calibration, did not exceed 1.5 km/h.

Keywords: Neural network, YOLO v3, Data for training the neural network (Dataset), Traffic flow assessment, Vehicle detection, Vehicle classification, Vehicle speed, Traffic monitoring

Introduction

Urbanization leads to a significant growth of the population density and road traffic concentration in large cities. This increased the likelihood of traffic accidents, road congestion, and led to increased vehicle emissions. In the conditions of urban infrastructural constraints, the tasks of ensuring an adequate population mobility can no longer be solved through the use of non-optimal heuristics based on a small amount of statistical information. Intelligent transport systems (ITS) of cities should ensure the maximum capacity of the road network and instantly respond to any traffic incidents to prevent road congestion. Currently, cities experience a rapid growth of video surveillance systems, which include video cameras with different resolutions and fixed frame rates with different resolutions and mounting points [1]. Continuous monitoring of quantitative

and qualitative road traffic parameters from fixed cameras will allow us to use vehicles as indicators of the transport system performance. The most reported issues when processing real-time data from street cameras are low counting accuracy, classification of a limited number of vehicle types, tracking an object with determining the speed and the driving direction in all sections when crossing the functional zone of the intersection. Despite the obvious advantages of developing such systems, there are few studies aimed at collecting and analyzing the speed and movement pattern of traffic flows through the use of survey street cameras [2]. Artificial neural networks have proven themselves to be good in the tasks of collecting, interpreting, and analyzing big data coming from video cameras [3].

Some studies [4] and [5] use low-resolution video surveillance system data and deep neural networks to count vehicles on the road and estimated traffic density. Examples of using conventional machine vision methods are systems developed in [6, 7], which analyzed the problems of freight traffic. To detect a vehicle, most modern works discuss the adaptation and improvement of modern detection systems, such as Faster R-CNN [8], YOLO [9], and SSD [10]. This includes architectural innovations solving the problems of scale sensitivity [11], vehicle classification [12–14], and increasing the speed and accuracy of the detection methods [15, 16]. Improving the detection rate [17], temporary information is also used for joint detection and tracking of objects [3, 18, 19].

The existing solutions in the problems of real-time vehicle detection and classification require large computing capabilities and place strict requirements for the installation location and camera performance.

Related work

Object detection

Neural network architectures can be conditionally divided into single-stage (RetinaNet, YOLO, SSD) and double-stage (R-CNN, Faster R-CNN, etc.) [20]. The main difference in these approaches is that the two-stage models generate regions at the first stage and classify at the second stage. This approach gives higher accuracy at the cost of the image processing speed. The single-stage approach generates and classifies at one stage, which provides a high image processing speed but lower accuracy. One of the main factors in this work is the image processing speed; therefore, to solve this problem, we considered single-stage networks.

To solve the problem of real-time object recognition, we considered the following neural networks: SSD, YOLO v3, RetinaNet, etc. After studying the performance tests [21, 22], we came to the conclusion that YOLO v3 shows the best result processing one image in 51 ms at a resolution of 608×608, which allows us to process 19 frames per second. Based on the real-time object detection task, the YOLO v3 neural network is capable to process the maximum number of frames per second, while it does not lose much in accuracy (Fig. 1).

An important feature of this architecture is that convolution layers are applied to the image once, unlike such architectures as R-CNN [23–25] and Faster R-CNN [8], which provides a multiple increase in the image processing speed without significant losses in accuracy: one image is processed 1000 times faster using YOLO than R-CNN, and 100 times faster than Fast R-CNN [24].

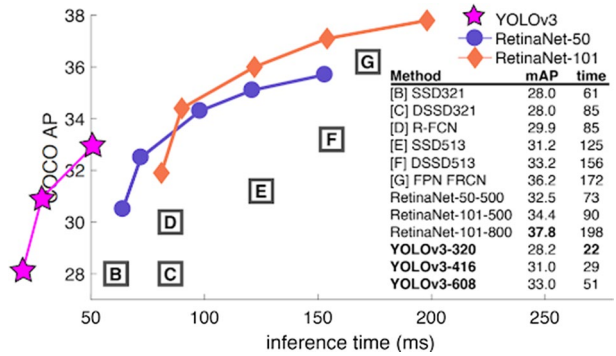

Fig. 1 Performance tests of neural networks on COCO dataset

Speed detection

The complexity of the task of determining the speed of vehicles on the video stream is caused by a large number of possible movement patterns, as well as the direction of the camera view center, which is not perpendicular to the movement patterns of vehicles. Several existing solutions are based on the use of traffic cameras located directly above the carriageway or on the side of it [26, 27]. In [28], the authors manually marked the measurement zone in the camera image. It is a rectangular area perpendicular to the traffic flow. In each frame, the Liang–Barsky algorithm checks the intersection of a vehicle with the measurement zone and counts the number of frames, over which the vehicle passed the measurement zone. Thus, the speed is defined as a ratio of the distance traveled to the travel time. In [29], the authors define the vehicle contour. Using the developed optical flow method, they determine the movement speed of the contour pixels. By adjusting the focal distance, angle, and height of the camera installation, the authors highlight the area of interest in the image so that it is equal to the width of the image. Thus, the vehicle speed (km/h) is calculated from the ratio between the image pixels and the road width.

The considered methods are focused on measuring the speed in preset zones with the known dimensions and traffic cameras located above the road and at a low height, which does not allow us to use them to collect data over the entire functional area of road junctions.

We propose a method to determine the average speed based on the coordinate mapping from the camera image to the space of geographic coordinates using a perspective transformation.

Methodology

The purpose of this work is to develop an autonomous approach able to assess the quantitative and qualitative parameters of road traffic, such as the amount, speed and movement pattern of vehicles. To this end, we divide the problem into four sub-tasks: detection and classification, tracking, counting and determining the average vehicle speed. This naturally leads to a modular and easily testable architecture consisting of indicator detection, tracking, and calculation modules. In the following sections, we will describe in detail each module together with the data collected for training and

assessment. Figure 2 shows an algorithm of obtaining the data on the driving directions and average speeds of vehicles.

The first module receives every third frame of the video stream and receives object predictions using YOLOv3. Upon receipt of the bounding boxes to find the average speed and determine the driving directions, the objects should be identified by comparison with the data from previous frames. To train the neural network, we collected a dataset from street surveillance cameras in Chelyabinsk. To track vehicles, we used the SORT tracker because it has a good compromise between speed and accuracy [30].

Detection of vehicles

Our approach is based on the use of static cameras with a viewing angle, which provides visibility of the entire physical territory of the intersection and adjacent roads. The camera angle was chosen with the condition of visibility of the entire physical territory of the intersection.

We used several freely accessed cameras of Intersvyaz company in the cities of Chelyabinsk [31] and Tyumen. We chose the cameras with a viewing angle providing visibility of the entire functional area of the intersection and adjacent roads. The cameras are located at a height of 14–40 m, with an elevation angle of 30–60° to the horizon. The video streams of these cameras provide a stable transmission of 25 frames per second, supporting a resolution of 1920×1080 pixels. At the same time, the video stream is not perfect due to compression artifacts, blurring, bad weather conditions, and hardware errors, which prevents the detection and classification of vehicles, as well as the determination of speed indicators using the existing methods.

We collected and tagged frames of video streams from 7 cameras of various road junctions as the data for training the neural network. As a result, we obtained about 6000 thousand images highlighting over 430,000 vehicle objects (Fig. 3).

The indexation of the classes and their corresponding colors further used to display the detection results are presented in Table 1.

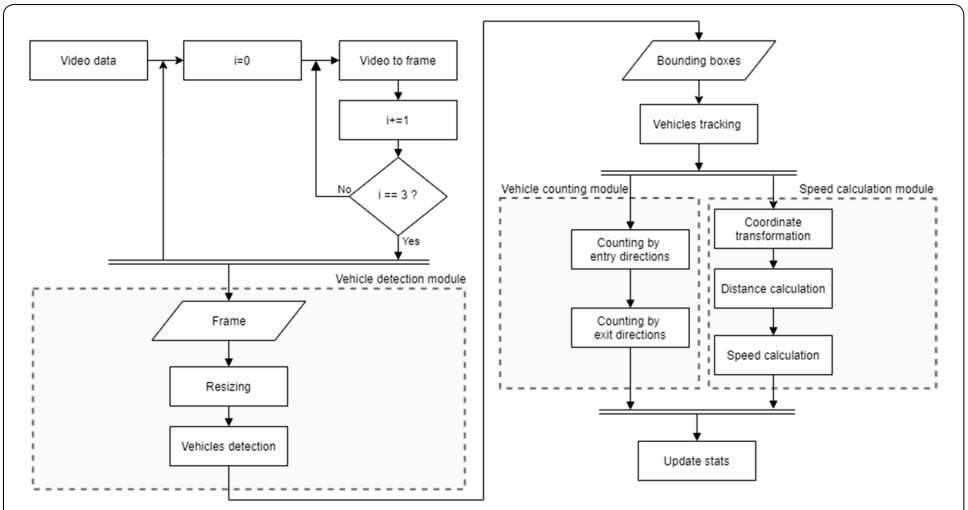

Fig. 2 An algorithm for determining the average speed and direction of vehicles

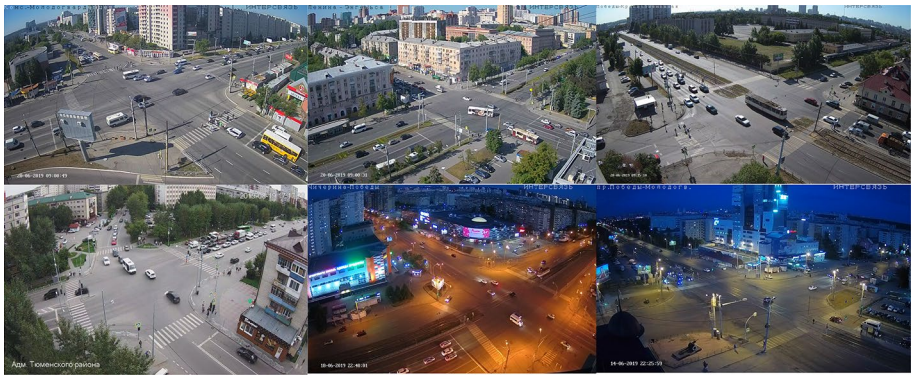

Fig. 3 Examples of an input image

Table 1 Indexation of the classes and their corresponding colors

Index	Class	Color
0	Car	Yellow
1	Mini_bus	Blue
2	Bus	Brown
3	Truck	Red
4	Tram	Pink
5	Trolleybus	Green

$$\begin{matrix} C_1 & X_1 & Y_1 & W_1 & H_1 \\ C_2 & X_2 & Y_2 & W_2 & H_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ C_i & X_i & Y_i & W_i & H_i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ C_n & X_n & Y_n & W_n & H_n \end{matrix}$$

Fig. 4 The input data

The input data are presented as follows: a JPG or PNG image and a text file with marking:

In Fig. 4. i is the object number; n is the number of objects in the image; C_i is the index of the class of the i-th object; X_i, Y_i are the coordinates of the center of the rectangle containing the object; W_i, H_i are the width and height of the rectangle containing the object.

The parameters X_i, Y_i, W_i, H_i are recorded in relative values of the image size ($X_i Y_i W_i H_i \in [0; 1]$).

For better training of the neural network, we expanded the dataset by applying augmentation, which increased the dataset by 10 times. For augmentation, we applied

Fig. 5 Augmented images

Table 2 Distribution of vehicle classes in the training sample

Class	Number of objects	Relation to the total number %
Car	3,518,370	81.8
Mini_bus	228,810	5.3
Bus	163,430	3.8
Truck	184,520	4.2
Tram	91,540	2.1
Trolleybus	112,690	2.6

the following transformations in various combinations: horizontal display; affine and perspective transformations; noise overlay; color distortion (Fig. 5).

The final dataset amounted to 4.3 million objects. The distribution of objects of each class in the training sample is presented in Table 2.

We divided the dataset into training and validation samples in the ratio of 80/20% and started training for 50,000 iterations with an increment of 0.001. The batch size per one iteration was 64 images, which was divided into 16 units during training to run several images at once.

Training of YOLOv3

The architecture of the YOLOv3 neural network consists of 106 layers (Fig. 6) and is a modification of the Darknet-53 neural network, which includes 53 layers (Fig. 7). Besides, it includes 53 more layers with two N-dimensional output layers, which allows us to make detections at three different scales. This modification contributes to a more accurate recognition of objects of various sizes. As input data, YOLOv3 accepts an image presented as a three-dimensional tensor of $h \times \times 3$, where h, are the height and length of the input image. The dimensionality of the output layers is determined by reducing the size of the input image by 32, 16, and 8 times, respectively (Fig. 6).

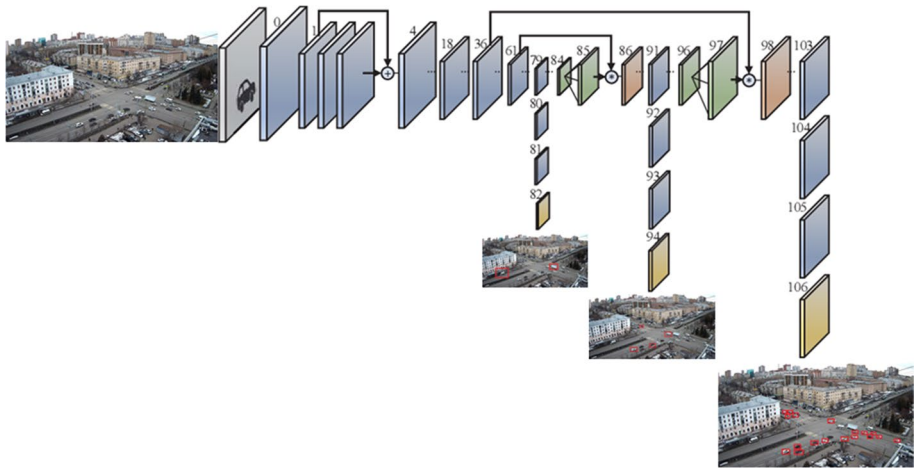

Fig. 6 The architecture of YOLO v3 [21]

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1 x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2 x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8 x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8 x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4 x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 7 The architecture of Darknet-53 [21]

In addition to the use of ultra-precise layers, its architecture YOLOv3 also contains residual levels [25], layers with increased discretization and passed connections. CNN takes the image as input data and returns a tensor (Fig. 8), which represents:

- coordinates and positions of the predicted bounding boxes, which should contain the objects;
- the probability that each bounding box contains an object;

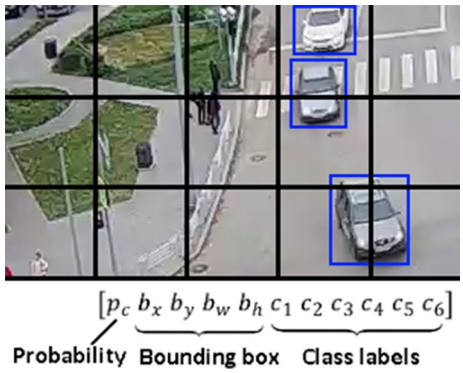

Fig. 8 Output tensor

- the probability that each object within its bounding box belongs to a certain class.

To train the YOLOv3 neural network, we used the backpropagation method with a gradient descent. This method is based on the use of the output error of a neural network to calculate the correction values for the weights of neurons in its hidden layers. The algorithm is iterative and uses the principle of training "by epochs", when the weights are changed after several instances of the training set are supplied to the neural network input, and the error is averaged for all the instances.

We improved the basic performance of YOLO with an additional mask branch and optimizing the shape of the anchors. An additional regression of the masks for each instance improves the precision in the corresponding regression problem of the bounding box. Consequently, the first optimization we applied was an additional mask branch. This branch runs in parallel with the existing branches and tends to regress the mask for each area of interest. For simplicity, we approximated the exact pixel masks of the instance using coarse polygonal masks from the collected dataset.

The results of training the neural network

The Average Precision (AP) is a popular indicator for measuring the precision of object detectors, such as Faster R-CNN, SSD, YOLOv3, etc. The calculate it, the AP values are used for each detected vehicle class, as shown in Fig. 9.

To obtain the "average precision" (mAP) for all classes, we average the AP values for each class. The average precision (mAP) of the system is 0.85.

Vehicle tracking

A comparison of the detected objects in the current frame with objects from the previous frames is a very difficult task. Vehicles detected in the previous frame may not be detected in the next frame for various reasons. For example, due to poor lighting conditions or occlusions, when one object is overlapped with another one. We solved the problem of multiple tracking of objects using the freely available SORT tracker. This is a simple and fast tracker operating in real time, which is very important in our task. It is based on two methods: the Kalman filter [32] and the Hungarian algorithm [33]. The linear speed is calculated for each object and the position of the object in the next

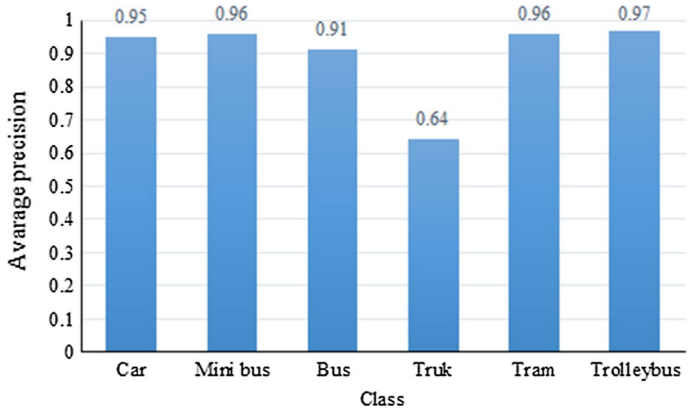

Fig. 9 Average precision values for classes

Fig. 10 The result of the tracker operation

frame is predicted. Based on the data received from YOLO, we calculated the shortest distance from each detected object to all the predicted ones. The Hungarian detection algorithm is used for the optimal matching of the predicted objects. Based on this data, the Kalman filter corrects the state of the object. The tracker assigns a unique identifier to each object (Fig. 10).

Each vehicle has its own identifier. To save memory and improve the tracking quality, the tracker takes into account an object only if it was detected at least in min_hits frames. If the object is not detected during max_age frames, it is deleted. In [34], the authors made a comparison using various metrics of several trackers operating in real time (RMOT, TDAM, MDP, etc.). As a result of the comparison, SORT showed the best ratio of speed and quality of operation: better or rather high indicators in the metrics of MOTA, MOTP, FP, FN, etc. at a frame rate of 260 on one Intel i7 2.5 GHz processor core and 16 GB of memory.

The video stream frequency of the camera is 25 frames per second. To increase the operating speed of the system, we skip every two frames and process only every third one. However, at some intersections, cars can drive at a high speed, abruptly change their movement pattern, and cannot be detected by the neural network in each frame due to poor lighting conditions, small size, or overlapping with tree branches. In such situations, the tracker may

not match all the new objects with the objects from the previous frames and assigns a new identifier to them. Therefore, we use a different number of passed frames for each intersection. Thus, between the frames, where the object was not detected, there will appear another frame, in which it can be detected. This allowed us to reduce errors at complex intersections, but at the same time increased the operating time.

Elimination of the camera distortion

Modern cameras are imperfect–they distort the image, changing the size, shape, and distances of objects. In our case, the image transmitted from the camera is subject to distortion. To determine accurately the coordinates of objects, we should eliminate the distortion by calibrating the camera. The easiest method of calibration is to use a spatial test object, such as a checkerboard [35], as shown in Fig. 11.

Figure 12 shows the source images and the images after applying the calibration.

Calculation of the distance

To calculate distance traveled, we must find the change in the latitude and longitude of the vehicle's location over a certain time interval using the change of coordinates in the camera image. To solve this problem, we calculated the perspective transformation matrix (Fig. 3) by selecting four reference points in the map and comparing the corresponding points in the image (Fig. 13).

To calculate the perspective transformation matrix $A = (c_{ij})_{3\times 3}$ we need to derive the coefficients c_{ij} from the following linear equations describing the dependence between the coordinates in the image and the geographic coordinates:

$$u_i = \frac{c_{00}x_i + c_{01}y_i + c_{02}}{c_{20}x_i + c_{21}y_i + c_{22}} \tag{1}$$

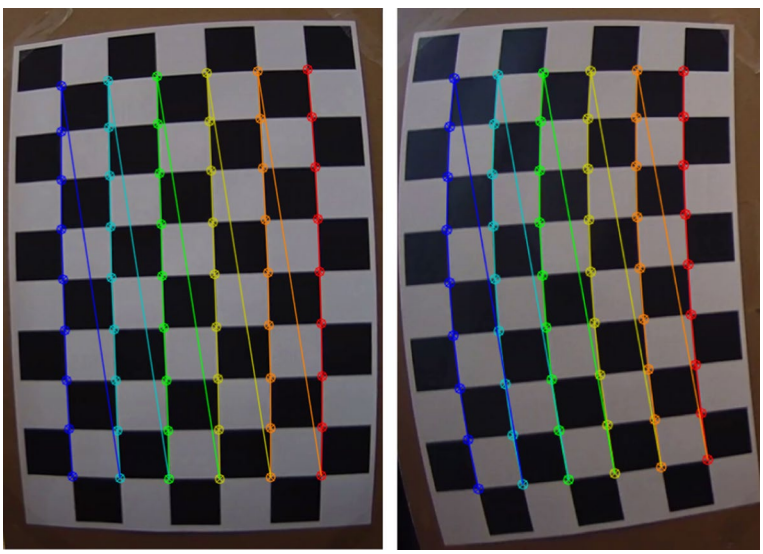

Fig. 11 Demonstration of correcting the image distortion through the use of a checkerboard

Fig. 12 Source and corrected camera images

Fig. 13 Reference points in the image

$$v_i = \frac{c_{10}x_i + c_{11}y_i + c_{12}}{c_{20}x_i + c_{21}y_i + c_{22}} \tag{2}$$

where u_i,v_i are geographic coordinates; c_{ij} are elements of the matrix A, $c_{22}=1$; x_i, y_i are the coordinates from the image, $i=1.4$.

As a result of the calculation, we solve the following matrix equation $(a_{ij})_{8\times 8} * (c_{ij})_{8\times 1} = (x_{ij})_{8\times 1}$ described in detail in Fig. 14.

After finding the matrix coefficients, we can perform transformation by multiplying the perspective transformation matrix by the coordinate vector from the image.

$$A \times \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x_i^{'} \\ y_i \\ t_i \end{pmatrix} \tag{3}$$

where A is the transformation matrix; x_i, y_i are the pixel coordinates in the image; x'_i, y'_i are the latitude and longitude of the point.

$$\begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0 \cdot u_0 & -y_0 \cdot u_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot u_1 & -y_1 \cdot u_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 \cdot u_2 & -y_2 \cdot u_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 \cdot u_3 & -y_3 \cdot u_3 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0 \cdot v_0 & -y_0 \cdot v_0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot v_1 & -y_1 \cdot v_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 \cdot v_2 & -y_2 \cdot v_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 \cdot v_3 & -y_3 \cdot v_3 \end{pmatrix} \times \begin{pmatrix} c_{00} \\ c_{01} \\ c_{02} \\ c_{10} \\ c_{11} \\ c_{12} \\ c_{20} \\ c_{21} \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Fig. 14 The matrix form of the solved equations

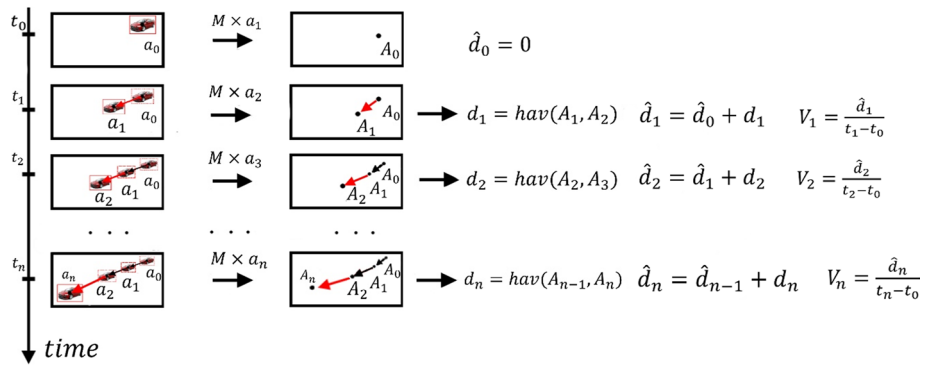

Fig. 15 The process for determining the distance and speed

To calculate the distance between two points, we find the distance between the two points on the sphere using the inverse haversine (4). The haversine in Eq. (4) is $h = ^2(\Theta/2)$. This method of determining the speed is universal for any movement pattern and does not require additional preliminary marking of the intersection and finding any reference distances.

$$d = rhav^{-1}(hav(\phi_2 - \phi_1) + \cos(\phi_2)\cos(\phi_1)hav(\lambda_2 - \lambda_1)) \tag{4}$$

where d is the measured distance; ϕ_1, ϕ_2, λ_1, λ_2 are the latitude and longitude of the i-th point; r is the radius of the earth ($r = 6371$ km).

Now, to calculate the average speed, we apply the following formula:

$$v = \frac{d}{t_2 - t_1} \tag{6}$$

where t_1, t_2 are the time of the beginning and end of movement at a distance.

To analyze the average speed of vehicles in real time, we record the time when

the vehicle appeared, as well as at each i-th step of receiving a frame from the video stream, we calculate the accumulated distance d_i used to find the average speed. The described algorithm is schematically shown in Fig. 15.

Fig. 16 Overview of intersections from CCTV cameras

Table 3 Counting of vehicles at four intersections

Data	Time	Intersections	Class					
			Car	Mini Bus	Bus	Truk	Tram	Trolley-bus
18.03.20	7:00–9:00	Komarova-Salyutnaya st.	5835/5896	315/313	22/25	21/20	0/0	0/0
18.03.20	17:00–19:00	Komarova-Salyutnaya st.	6912/6767	218/227	18/19	11/11	0/0	0/0
17.03.20	7:00–9:00	Pobedy-Molodogvardeit-sev st.	6587/6677	382/364	19/21	23/23	49/48	0/0
17.03.20	17:00–18:00	Chicherina-Pobedy st.	6210/6178	259/246	48/47	13/12	50/52	9/9
18.03.20	7:00–9:00	Pobedy-Krasnoznamen-naya st.	4501/4554	228/239	41/40	57/54	73/73	0/0
18.03.20	17:00–19:00	Komsomolskiy-Sverdlovskiy st.	9865/9501	601/602	43/41	6/6	0/0	109/102

In Fig. 15: is the perspective transformation matrix, a_i are the coordinates of a specific vehicle, d_i is the distance between two points, t_i is the time between frames, v_i is the vehicle speed at the section d_i, V is the average speed.

Updating the data on the average vehicle speed when processing each frame of the video stream allows us to use the proposed method in real time.

Experimental results and discussions

Counting the vehicles

To assess the counting quality, we took the video content from CCTV cameras lasting from 1 to 2 h. We performed preliminary preparation for each intersection: marking the driving direction, a mask hiding parking spaces and the adjacent territory (Fig. 16).

Table 3 shows the values of the programmed and manual vehicle counting at the intersections of the city of Chelyabinsk.

Table 4 Counting errors

Error (%)	Car	Mini bus	Bus	Truck	Tram	Trolley-bus
Mean	1.6	3.2	13.6	2.9	2.0	3.2
Max	3.6	5.0	6.4	7.6	4.0	6.4

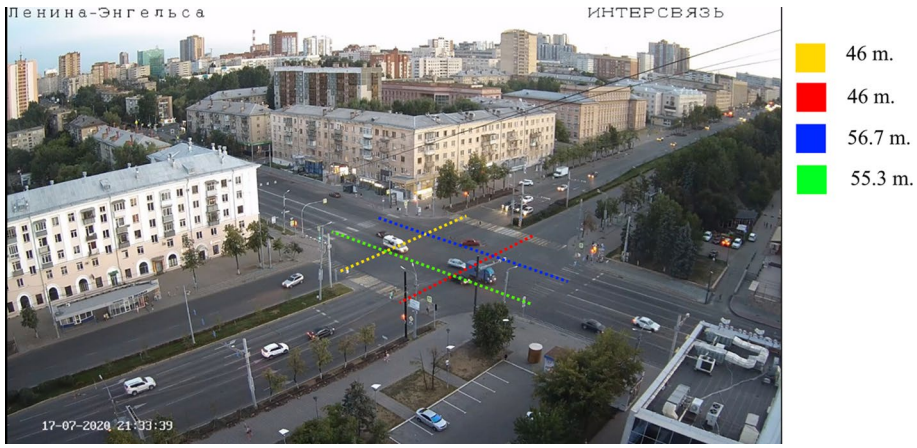

Fig. 17 Measured movement patterns and their lengths

Table 4 shows the percentage of the counting error for each class. After analyzing the data of manual and programmed vehicle counting, we found out that the mean counting error for all the classes is 5.5% of the total number of vehicles.

An additional study of typical errors showed that most of them result from strong and prolonged occlusions between vehicles in queuing traffic. For example, while a trolleybus or a truck is moving, one or two lanes are partially blocked. Many cars are overlapped when turning, waiting in the center of the intersection for a free window. This problem can be solved by improving the tracking module using special methods for instance re-identification based on appearance tips. However, as it has been mentioned above, the existing approaches have a high computation load and are not applicable to real systems. The development of efficient algorithms for re-identification of vehicles remains an open question.

Average vehicle speed

We conducted comparative testing to check the accuracy of the proposed system. To this end, we made manual calculations of the average vehicle speed. Namely, the travel time of the vehicle was measured on movement patterns with a priori known distances (Fig. 17).

This video was processed by the program, a comparison with the program calculation result is presented in Table 5.

As a result of analyzing the obtained data, we revealed the maximum speed determination error of 1.5 km/h, the mean error for all the movement patterns is 0.57 km/h.

Table 5 The experimental results of the speed detection system

Vehicles	Direction 1 yellow		Direction 2 red		Direction 3 blue		Direction 4 green	
	Real	Detection	Real	Detection	Real	Detection	Real	Detection
1	48.7	48.1	59.1	60.2	34.4	35.7	31.9	31.7
2	31.8	32.3	43.6	44.2	34.2	34.0	34.7	35.1
3	53.4	53.1	44.8	44.4	29.2	28.5	27.9	27.6
4	40.3	41.0	40.4	40.6	40.2	40.4	40.4	40.0
5	63.7	64.3	55.3	55.3	33.1	32.9	30.0	31.5
6	53.4	53.5	38.5	39.7	35.3	36.4	36.4	37.3
Max	0.7		1.2		1.3		1.5	
Mean	0.46		0.58		0.62		0.62	

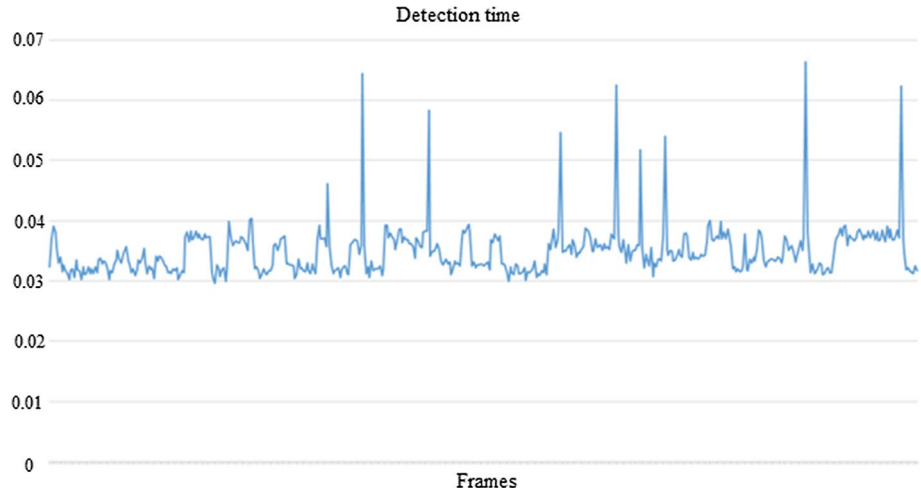

Fig. 18 The time spent on vehicle detection for one frame

Time complexity

So that the proposed method for determining the speed and number of vehicles could work in real time, it is necessary that the time complexity for processing each frame did not exceed $1/q$, where q is the number of frames per second. For the test intersection, we used every third frame of the video stream; therefore, the upper estimate of the time complexity of processing one frame will be $1/25 \times 3{=}0.12$.

Figures 18, 19 show the time complexities for the vehicle detection and speed calculation processes.

The tests were made on a PC with the following specifications: CPU: i9 9900 k, GPU: GeForce RTX 2080TI, RAM: 64 GB. The maximum time spent on vehicle detection for one frame was 0.066 s, the maximum time for calculating the speed and counting was 0.009 s. In addition to the main processes implementing the above methodology, the software solution consists of many auxiliary processes responsible for data transfer, aggregation, and storage. Figure 20 shows a diagram of the time spent to complete all the processes and obtain the final data for the tested intersection.

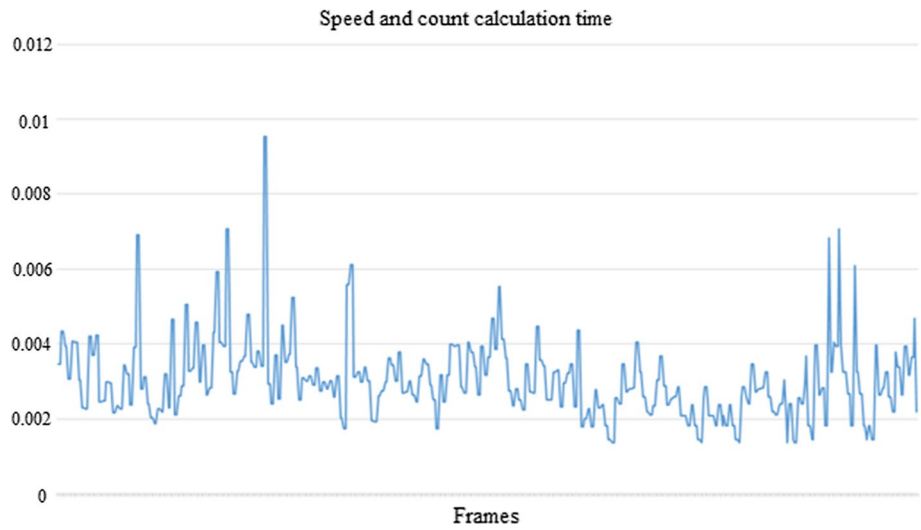

Fig. 19 The time spent for calculating the speed and number of vehicles for all the directions

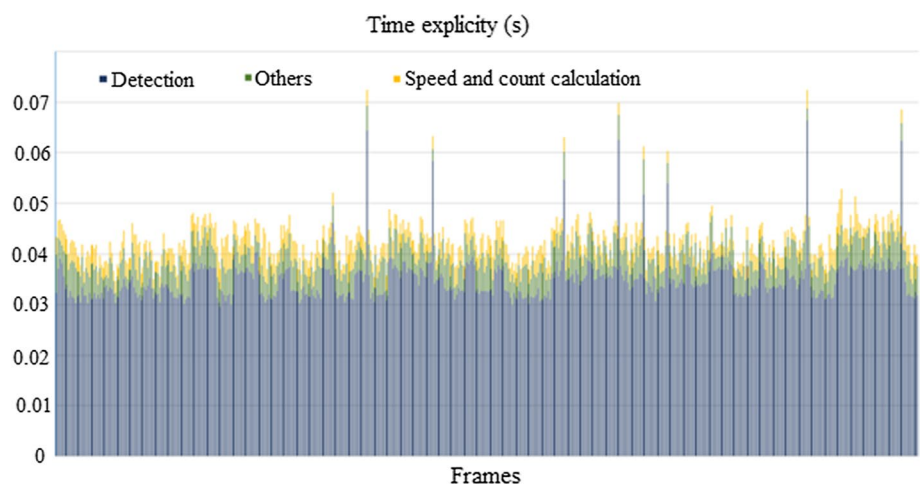

Fig. 20 The time of complete processing of one frame

After analyzing the data, we can conclude that the upper estimate of the time complexity of processing one frame is 0.08 s, which fits into the above limitations and allows us to use the presented method to determine the speed and monitor traffic in real time.

Software solution

The system includes the following sequence of processes (Fig. 21):

- frames reading (Process 1);
- detection and classification of vehicles from the current frame (Process 2);
- vehicle tracking and counting in all directions of the road junction (Process 3);
- calculation of the latitude and longitude of the vehicle location (Process 4);
- calculation of vehicle speeds (Process 5);

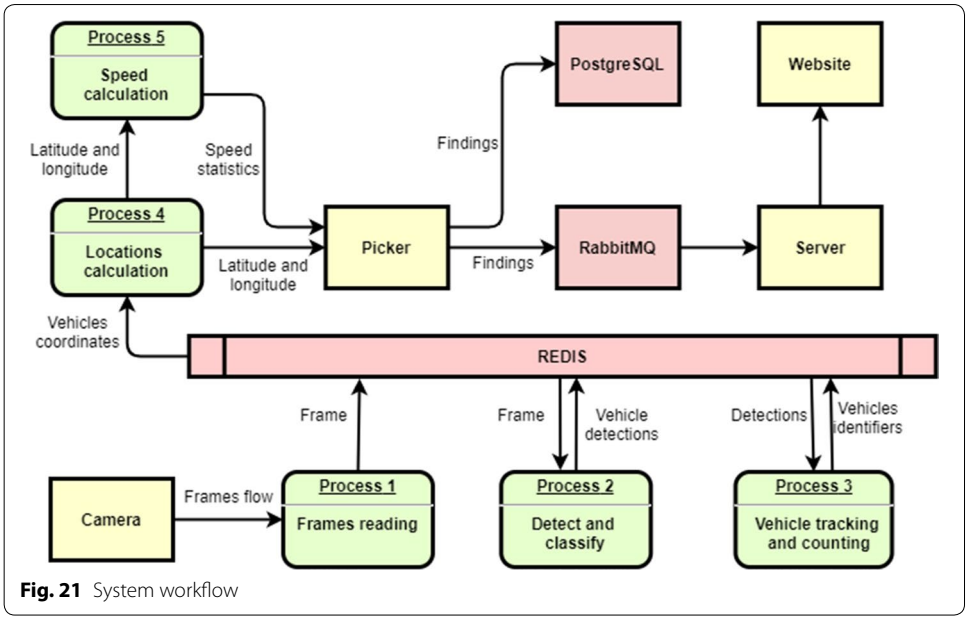

Fig. 21 System workflow

- calculation of metrics related to the amount of harmful substances emitted by each vehicle (Process 6).

Used technologies

We used the following technologies for the software implementation of the presented architecture:

1. OpenCV is an open-source library designed to work with computer vision algorithms, image processing and general-purpose numerical algorithms. We used this library to perform the following tasks:

   a. Resizing an image and applying a mask to it;
   b. Setting and displaying of entry and exit areas, as well as determining the presence of vehicles in said areas;
   c. Camera calibration and elimination of distortion;
   d. Use of the perspective transformation matrix and determining the length of the distance;
   e. Data visualization.

2. Sort is an open-source library for 2D tracking of several objects in video sequences based on the elementary data association and state estimation methods. We used it to track vehicles in the video stream.
3. Redis is a resident open-source NoSQL-class database management system. We used it to store intermediate results of the modules.
4. RabbitMQ is a software message broker based on the AMQP standard. We used it to organize a data queue for transferring to a web page.

5. PostgreSQL is a free object-relational database management system. To compile statistics and calculate various metrics, such as KPI and daily flow structure, we aggregate and save the received data in a database every hour.

Conclusion

In this study, we focused on the problem of obtaining the data on the speed and driving direction of vehicles based on the video stream from street surveillance cameras. The complexity of the task is caused by the following factors: different viewing angle, remoteness from the intersection, overlapping of objects. We added an additional mask branch in the YOLO v3 neural network architecture and optimized the shapes of anchors to improve the accuracy of detection and classification of objects of different sizes to improve the quality of object tracking. To determine the speed in real time, we presented a method based on the application of a perspective transformation of the coordinates of vehicles in the image to geographic coordinates.

The proposed system was tested at night and in the daytime at six intersections in the city of Chelyabinsk, showing a mean vehicle counting error of 5.5%. The error in determining the vehicle speed by the projection method, taking into account the camera calibration at the tested intersection, did not exceed 1.5 m/s. The presented methodology allows us to generate complete and high-quality data for real-time traffic control and significantly reduce the requirements to peripheral equipment. Within the framework of this study, we did not consider the solution of many problems, such as overlapping of objects, a more detailed classification of vehicles, the definition of accidents and blocking objects. We consider our solution as a basis for our future research aimed at solving these problems.

Abbreviations

ITS: Intelligent transport systems; CNN: Convolutional neural networks; AP: Average precision; CCTV: Closed-circuit television.

Acknowledgements

Not applicable.

Authors' contributions

VS and KH designed research, performed research, analyzed the data, and wrote the paper. IS and TC designed research and was a major contributor in writing the manuscript. IC, IA and SS gathered data and wrote the paper. All authors suggested related works, discussed the structure of the paper and results. All authors read and approved the final manuscript.

Funding

The work was supported by Act 211 Government of the Russian Federation, contract No. 02.A03.21.0011.

Availability of data and materials

https://github.com/Readix/TrafficMonitoring

Competing interests

The authors declare that they have no competing interests.

Received: 7 May 2020 Accepted: 11 September 2020
Published online: 06 October 2020

References

1. Peppa MV, Bell D, Komar T, Xiao W. Urban traffic flow analysis based on deep learning car detection from cctv image series. Int Arch Photogramm Remote Sens Spat Inf Sci. 2018;42(4):565–72. https://doi.org/10.5194/isprs-archives-XLII-4-499-2018.

2. Fedorov A, Nikolskaia K, Ivanov S, Shepelev V, Minbaleev A. Traffic flow estimation with data from a video surveillance camera. J Big Data. 2019. https://doi.org/10.1186/s40537-019-0234-z.
3. Li C, Dobler G, Feng X, Wang Y. TrackNet: simultaneous object detection and tracking and its application in traffic video analysis. 2019; pp. 1–10. arxiv.org/pdf/1902.01466.pdf.
4. Zhang F, Li C, Yang F. Vehicle detection in urban traffic surveillance images based on convolutional neural networks with feature concatenation. Sensors. 2019;19(3):594. https://doi.org/10.3390/s19030594.
5. Zhang S, Wu G, Costeira JP, Moura JM. FCN-rLSTM: Deep spatio-temporal neural networks for vehicle counting in city cameras. In: Proceedings of the IEEE international conference on computer vision. 2017. https://doi.org/10.1109/iccv.2017.396.
6. Rathore MM, Son H, Ahmad A, Paul A. Real-time video processing for traffic control in smart city using Hadoop ecosystem with GPUs. Soft Comput. 2018;22(5):1533–44. https://doi.org/10.1007/s00500-017-2942-7.
7. Sun X, Ding J, Dalla Chiara G, Cheah L, Cheung NM. A generic framework for monitoring local freight traffic movements using computer vision-based techniques. In: 5th IEEE international conference on models and technologies for intelligent transportation systems (MT-ITS). 2017. p. 63–8. https://doi.org/10.1109/mtits.2017.8005592.
8. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017;39(6):1137–49. https://doi.org/10.1109/TPAMI.2016.2577031.
9. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). 2016. https://doi.org/10.1109/cvpr.2016.91.
10. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. SSD: single shot multibox detector. Lect Notes Comput Sci. 2016;9905:21–37. https://doi.org/10.1007/978-3-319-46448-0_2.
11. Hu X, Xu X, Xiao Y, Chen H, He S, Qin J, Heng PA. SINet: a scale-insensitive convolutional neural network for fast vehicle detection. IEEE Trans Intell Transp Syst. 2019;20(3):1010. https://doi.org/10.1109/TITS.2018.2838132.
12. Jung H, Choi MK, Jung J, Lee JH, Kwon S, Jung WY. ResNet-based vehicle classification and localization in traffic surveillance systems. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW). 2017. 934–40. https://doi.org/10.1109/cvprw.2017.129.
13. Li S, Lin J, Li G, Bai T, Wang H, Pang Y. Vehicle type detection based on deep learning in traffic scene. Procedia Comput Sci. 2018;131:564–72. https://doi.org/10.1016/j.procs.2018.04.281.
14. Sommer L, Acatay O, Schumann A, Beyerer J. Ensemble of two-stage regression based detectors for accurate vehicle detection in traffic surveillance data. 2019. p. 1–6. https://doi.org/10.1109/avss.2018.8639149.
15. Wang L, Lu Y, Wang H, Zheng Y, Ye H, Xue X. Evolving boxes for fast vehicle detection. In: 2017 IEEE international conference on multimedia and Expo (IC-ME). 2017. p. 1135–40. https://doi.org/10.1109/icme.2017.8019461.
16. Zhu F, Lu Y, Ying N, Giakos G. Fast vehicle detection based on evolving convolutional neural network. In: 2017 IEEE international conference on imaging systems and techniques (IST). 2017. p. 1–4. https://doi.org/10.1109/ist.2017.8261505.
17. Anisimov D, Khanova T. Towards lightweight convolutional neural networks for object detection. In: 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS). 2017; 1–8. https://doi.org/10.1109/avss.2017.8078500.
18. Li S. 3D-DETNet: a single stage video-based vehicle detector. 2018. arxiv.org/ftp/arxiv/papers/1801/1801.01769.pdf.
19. Luo W, Yang B, Urtasun R. Fast and furious: real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. 2018; 3569–77. https://doi.org/10.1109/cvpr.2018.00376.
20. Wu Y, Jiang S, Xu Z, Zhu S, Cao D. Lens distortion correction based on one chessboard pattern image. Front Optoelectron. 2015;8(3):319–28. https://doi.org/10.1007/s12200-015-0453-7.
21. Redmon, J., Farhadi, A. YOLOv3: An Incremental Improvement. 2018. arxiv.org/pdf/1804.02767.pdf
22. Lin T-Y, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. 2017. arxiv.org/pdf/1708.02002.pdf
23. He K, Gkioxari G, Dollar P, Girshick R. Mask R-CNN. In: 2017 IEEE international conference on computer vision (ICCV). vol. 2017: 2017; 2980–8. https://doi.org/10.1109/iccv.2017.322.
24. Shreyas Dixit KG, Chadaga MG, Savalgimath SS, Ragavendra Rakshith G, Naveen Kumar MR. Evaluation and evolution of object detection techniques YOLO and R-CNN. Int J Recent Technol Eng. 2019;8(3):824–9. https://doi.org/10.35940/ijrte.B1154.0782S319.
25. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 770–8. https://arxiv.org/pdf/1512.03385.pdf.
26. Javadi S, Dahl M, Pettersson MI. Vehicle speed measurement model for video-based systems. Comput Electr Eng. 2019;76:238–48. https://doi.org/10.1016/j.compeleceng.2019.04.001.
27. Gholami A, Dehghani A, Karim M. Vehicle speed detection in video image sequences using CVS method. Int J Phy Sci. 2010;5(17):2555–63.
28. de Barth O VB, Oliveira R, de Oliveira MA, Nascimento VE. Vehicle speed monitoring using convolutional neural networks. IEEE Latin Am Trans. 2019;17(06):1000–8. https://doi.org/10.1109/tla.2019.8896823.
29. Lan J, Li J, Hu G, Ran B, Wang L. Vehicle speed measurement based on gray constraint optical flow algorithm. Optik Int J Light Elect Optics. 2014;125(1):289–95. https://doi.org/10.1016/j.ijleo.2013.06.036.
30. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. Simple online and realtime tracking. In: 2016 IEEE international conference on image processing (ICIP). 2016: 3464–8. https://doi.org/10.1109/icip.2016.7533003.
31. Video observation. https://cams.is74.ru/live. Accessed 20 May 2020.
32. Kalman R. A new approach to linear filtering and prediction problems. J Basic Eng. 1960;82:35–45. https://doi.org/10.1115/1.3662552.
33. Kuhn HW. The Hungarian method for the assignment problem. Naval Res Log Quart. 1955;2:83–97. https://doi.org/10.1002/nav.3800020109.

34. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). 2016. 3464–8. https://doi.org/10.1109/icip.2016.7533003.
35. Wu W, Wu L, Li J, Wang S, Zheng G, He X. RetinaNet-based visual inspection of flexible materials. In: 2019 IEEE International Conference on Smart Internet of Things (SmartIoT). 2019; 432–5. https://doi.org/10.1109/smartiot.2019.00077.

Publisher's Note