Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

An accurate approach for obtaining spatiotemporal information of vehicle loads on bridges based on 3D bounding box reconstruction with computer vision

Jinsong Zhu [a,*], Xingtian Li [a,b], Chi Zhang [a], Teng Shi [a]

[a] Key Laboratory of Coast Civil Structure Safety (Ministry of Education), School of Civil Engineering, Tianjin University, Tianjin, China
[b] School of Civil Engineering, Lanzhou Jiaotong University, Lanzhou, China

ARTICLE INFO

Keywords:
Bridge
Computer vision
Spatiotemporal information
Vehicle loads
Weighing

ABSTRACT

Vehicle load is an important basis for bridge design, condition assessment, and maintenance and strengthening, and its statistical laws can help us further understand the behavior of bridges under vehicle loads. Therefore, it is important to obtain accurate vehicle weight and vehicle spatiotemporal information that reflects the load time history. As of now, vehicle weight can be obtained using WIM technology, but there are still some issues in the methods of obtaining vehicle information. The coordinate transformation method is simple to operate, but sacrifices accuracy. The existence of a certain distance from the tail of vehicles to bridge decks makes the results based on dual-target detection differ from the actual. Therefore, an accurate approach for obtaining spatiotemporal information of vehicle loads on bridges based on 3D bounding box reconstruction with computer vision is proposed in this paper. To achieve this, a deep convolutional neural network (DCNN) and the You Only Look Once (YOLO) detector are used to detect vehicles and get the 2D bounding box. By establishing the relationship between 2D and 3D bounding box of the vehicle, an algorithm for 3D bounding box reconstruction of vehicles is proposed to get the sizes and position of vehicles. The spatiotemporal information of the vehicle loads is finally obtained by using multiple objects tracking (MOT). To verify the accuracy and reliability of the proposed approach, a bridge vehicle loads identification system (BVLIS) was developed and tested on a cable-stayed bridge in operation. The results show that the approach is accurate and reliable, and can be used to obtain vehicle information and provide vehicle load boundary conditions for bridge finite element modeling.

1. Introduction

As a major variable load on bridges, vehicle loads may lead to fatigue cracking or even collapse of bridges. In addition, vehicle loads are also an important basis for bridge design, condition assessment, safety evaluation, and maintenance and strengthening, and the statistical laws can help us further understand the behavior of bridges under vehicle loads. Therefore, an accurate and reliable identification of vehicle loads is very important, which includes vehicle weighing and the acquisition of vehicle spatiotemporal information (i.e., type, position, size, speed and trajectory, etc.).

The early methods to obtain gross vehicle weight (GVW) were static weighing techniques [1], which had good accuracy, but were also expensive and time-consuming. For weighing while the vehicle is in motion, weigh-in-motion (WIM) techniques [2–4] were proposed. However, the corresponding equipment is subject to damage from heavy vehicles and is less durable. Later, the concept of bridge weigh-in-motion (B-WIM) [5] was proposed and several B-WIM systems (e.g., Axway, Culway, and SiWIM) [6–8] were developed to obtain wheel loads. Nonetheless, it is difficult to obtain accurate results due to the effect of vehicle lateral position and vibration [9]. Therefore, the moving force identification (MFI) methods for obtaining the time history of moving loads were proposed, including the interpretive method I (IMI) [10], the interpretive method II (IMII) [11], the time domain method (TDM) [12], and the frequency-time domain method (FTDM) [13]. Meanwhile, the researches in data analysis [14] and sensing technology [15–18] were also done to improve the accuracy. However, the application of dynamic algorithms to practice is difficult due to the large amount of numerical calculation and the high demand on the finite element model.

In recent years, the computer vision technology has been introduced into the monitoring of infrastructure [19–22]. By using cameras for

* Corresponding author.

https://doi.org/10.1016/j.measurement.2021.109657
Received 5 March 2021; Received in revised form 10 May 2021; Accepted 23 May 2021
Available online 29 May 2021

bridge deflections measurement and traffic monitoring, Ojio [23] proposed a contactless bridge weigh-in-motion (cBWIM) system. Feng et al. [24] introduced an innovative weighing method by estimating the contact pressure of a vehicle tire and the contact area, but the lack of tire pressure data makes it difficult to use in practice. In order to obtain spatiotemporal information such as vehicle location and trajectory, scholars have performed real-time information acquisition by vehicle detection (using background subtraction or Gaussian mixture model method) and tracking (using template matching, particle filtering techniques, or Kalman filtering algorithms) [25–26]. However, the sensitivity of detection algorithms to environmental conditions (e.g., lighting and shadows) makes the vehicle information acquisition method less robust.

With the rapid development of deep learning techniques and the availability of well-established detectors, vehicle detection has become efficient and reliable, and some deep learning-based methods for obtaining vehicle information have appeared in related reports. Zhang et al. [27] proposed a method of getting spatiotemporal information of vehicles on bridges based on the technology of DCNN and image calibration method. However, the image calibration method requires standard vehicles for data acquisition and fitting, which undoubtedly increases the complexity of system deployment. Zhou et al. [28] proposed an identification method in which the vehicle was detected by using a trained Faster R-CNN model and then tracked using the Kalman filter method. Jian et al. [29] proposed a traffic sensing methodology capable of automatically identifying the vehicle loads and speeds. Xia et al. [30] proposed a traffic monitoring methodology for complicated traffic scenarios. Although the coordinate transformation method makes the mapping of information in images to 3D space simple and easy, it introduces more computational errors as a result. Ge et al. [31] proposed a monitoring method of full-bridge traffic load distribution based on YOLO-v3 machine vision. In this report, the spatial information of the vehicle is obtained using a dual-target detection model that can detect the profile and the tail of vehicles and the vision principle. However, the distance between the tail of the vehicle and the bridge deck in reality will inevitably bring errors to the calculation of spatial information. Meanwhile, the workload of image annotation has increased significantly compared to single-target detection.

Although the WIM techniques can obtain some vehicle information, its accuracy is not enough. Therefore, scholars have turned to computer vision techniques to obtain spatiotemporal information about vehicles. Since the traditional image recognition methods have high sensitivity to environmental noise, it is difficult to cope with complex scenes with multiple vehicles. To address this issue, deep learning techniques are used for vehicle detection. Combining vehicle detection and tracking, some methods for vehicle spatiotemporal information acquisition are proposed in the latest reports. However, there are still some issues. The image calibration method does not reflect the real situation in 3D space; The coordinate transformation method is simple and easy, but at the cost of loss of accuracy; The method based on dual-target detection cannot correctly obtain the distance from the tail of the truck to the bridge deck. The presence of these factors, together with the distant traffic cameras, makes more inaccurate in the acquired vehicle spatiotemporal information, which affects our assessment of the bridge condition and our perception of its behavior. Therefore, we try to combine deep learning-based vehicle detection, camera calibration, and reconstruction of vehicle 3D bounding boxes to obtain the accurate vehicle spatiotemporal information. The innovation of this paper is the establishment of the vehicle 3D bounding box reconstruction model and the derivation of the algorithm equations, as well as the accurate vehicle load spatiotemporal information thus obtained. This approach eliminates excessive errors caused by assumptions and approximations, thus making the obtained data more reliable and practical.

In this paper, we proposed an accurate approach for obtaining spatiotemporal information of vehicle loads on bridges based on 3D bounding box reconstruction with computer vision. The 2D bounding box of the vehicle is first obtained using deep learning technology. Then the 3D bounding box was reconstructed based on the reconstruction model by constructing spatial mapping relationship and the vehicle sizes and position were obtained. Finally, the vehicle spatiotemporal information was obtained by using MOT. The rest of the paper consists of two parts: The first part is a statement of principle and the second part is an application of the approach. Section 2 explains the research framework in detail. Section 3 shows the DCNN architecture and detector framework, then explains the process of the dataset creation and gives the loss function in the vehicle detection model. In Section 4, an approach for reconstructing 3D bounding box of vehicles is proposed by constructing spatial mapping relationship between 2D and 3D bounding box. In Section 5, the tracking and spatiotemporal information of vehicles is obtained using MOT. Section 6 gives an application of the proposed approach and analyzes the error of the test results. Conclusions are given in Section 7.

2. The framework

The framework contains 3 parts: The first part is detection and classification of vehicles. Part two contains 'Bridge-Vehicle-Camera' system (BVCS) and the reconstruction model of vehicle 3D bounding box. The last part is vehicle tracking. The details are shown in Fig. 1.

In part one, the images of vehicles are captured on urban bridges or roads using cameras in a variety of different environments (e.g. different lighting conditions). Then, the type and axis-aligned ground truth box of vehicles are annotated, and the dataset is formed. Considering the speed and accuracy in real-time detection, YOLO-v4, gradient descent algorithm and hardware acceleration are used for the model training. When the loss function drops rapidly and stabilizes, the weight and bias are saved and the vehicle detector is obtained. Finally, the detector is used to detect different types of vehicles, and the confidence score and 2D bounding box are obtained.

In part two, a camera is installed on the side of the bridge deck, thus forming the BVCS. Then, the camera is calibrated with two different calibration boards. A small calibration board is used to obtain the accurate intrinsic parameter matrix. The extrinsic parameter matrix is then obtained using a large calibration board laid on the bridge deck. By establishing the relationship between 2D and 3D bounding box, a model for 3D bounding box reconstruction is proposed. Finally, the confidence score and 3D bounding box are obtained.

In the last part, a region of interest (ROI) is set on the bridge deck firstly. Then, the vehicle status matrix is formed during detecting. According to threshold value based on estimates and field tests, vehicle tracking is achieved by traversing all vehicles in the previous frame to get the distance between vehicles in two adjacent frames. Once a vehicle enters the ROI, its trajectory, No. and the moment are recorded and the status matrix is updated. The axle number is then determined based on vehicle type, size and statistics. When the vehicle leaves the ROI, the tracking is terminated and the information is written to a data file.

3. Vehicle detection

3.1. YOLO-v4 framework

As the vehicle travels at high speeds on the bridge, the real-time detection capability is critical. Based on DCNN, several detectors have been proposed for object detection, including two-stage (RCNN, SPPNet, Fast RCNN, Faster RCNN) and one-stage (DetectorNet, OverFeat, YOLO and SSD) detectors [32–35]. In comparison, the two-stage detectors are computationally expensive for many devices with limited storage and computational capability, the one-stage detector provides a good balance between efficiency and accuracy. Being one-stage detectors, both SSD and YOLO have good detection performance. However, with the improvement of algorithm, the latest version of YOLO detector [35] has obvious advantages in accuracy and speed. Therefore, YOLO-v4 is used

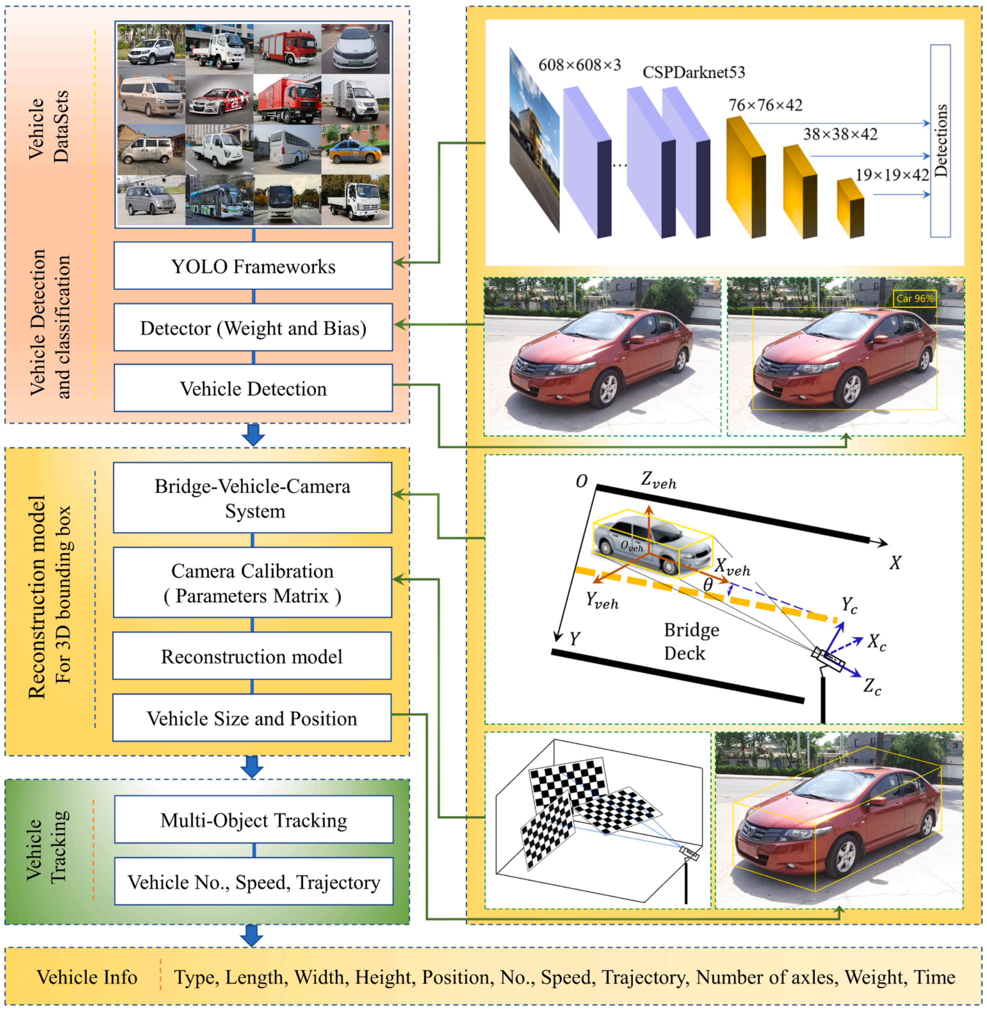
Vehicle DataSets
Vehicle Detection and classification
YOLO Frameworks
Detector (Weight and Bias)
Vehicle Detection
Reconstruction model For 3D bounding box
Bridge-Vehicle-Camera System
Camera Calibration (Parameters Matrix)
Reconstruction model
Vehicle Size and Position
Vehicle Tracking
Multi-Object Tracking
Vehicle No., Speed, Trajectory
608×608×3
CSPDarknet53
76×76×42
38×38×42
19×19×42
Detections
Car 96%
O
Z_{veh}
X
O_{veh}
X_{veh}
θ
Y_{veh}
Y_c
X_c
Z_c
Y
Bridge Deck
Vehicle Info
Type, Length, Width, Height, Position, No., Speed, Trajectory, Number of axles, Weight, Time

Fig. 1. Framework of the proposed approach (Note: Vehicle datasets is used for model training in deep learning. Detector refers to the data file containing the weights and biases after the model is trained and the program code used for vehicle detection. When the detector reads in the vehicle image, it extracts the feature map and plots the detection results as shown in the graph on the right. Based on vehicle detection, camera calibration, etc., the reconstruction model is proposed and used for the acquisition of vehicle size and spatial position. In order to get information such as the speed and trajectory of the vehicle, the vehicle must be tracked.)

for vehicle detection.

As shown in Fig. 2, the framework of YOLO-v4 consists of 5 parts: Input, backbone, neck, prediction and output. To achieve an optimal balance in terms of the input network resolution, the convolutional layer number and the parameter number, the CSPDarknet53 with 53 convolutional layers and 23 residual blocks is used as the backbone of the

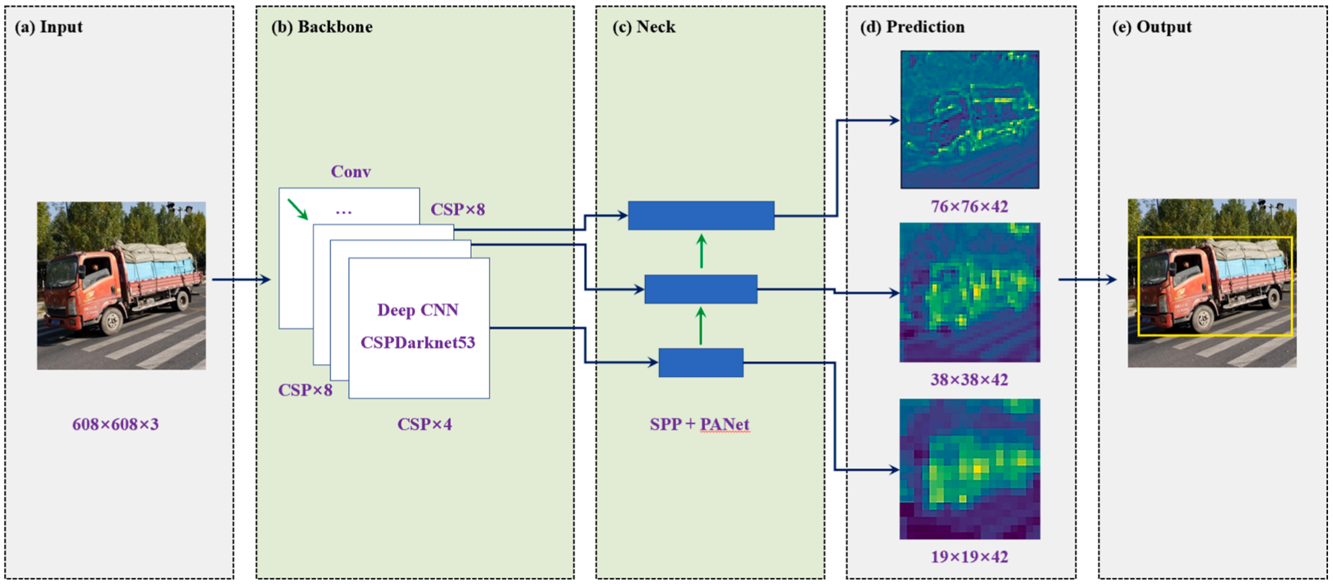

Fig. 2. YOLO-v4 framework (Note: The CSP refers to Cross-Stage-Partial strategy, and the number 8 after it means that the block contains 8 residual blocks. The Neck contains 3 operation modules for further processing the feature maps output from CSPDarknet53 to obtain richer spatial information.)

YOLO-v4. The CSPDarknet53 is the network formed by applying the Cross-Stage-Partial (CSP) strategy to Darknet53, and its architecture are listed in Table 1. When the image with 608 × 608 pixels is passed to CSPDarknet53, feature maps at different scales are obtained. In order to make the feature maps contain richer information and thus the framework has better detection accuracy, two techniques are introduced into it, one is spatial pyramid pooling (SPP) module and the other is path aggregation network (PANet). The SPP module significantly increased the receptive field and separates the most important contextual features without slowing down the network operation. The PANet introduced a short-cut path to make low-layer information easier to propagate to the top and make fine-grain localized information available to top layers. In the prediction stage, the multi-class classification and bounding box regression are carried out on multiple scales. In order to measure the distance between the prediction and the actual and to optimize the weights, the value of the loss function needs to be calculated. The loss function in YOLO-v4 consists of 3 parts: Bounding box regression loss, classification loss, and confidence loss, whose expressions are as follows:

$$Loss = L_{loc} + L_{cls} + L_{conf}. \tag{1}$$

In bounding box regression, the mean square error (MSE) method used in the old version of YOLO is removed and replaced by the complete-IoU (CIoU) algorithm. This is because the CIoU loss considers several aspects of overlapping area, center-of-mass distance and aspect ratio simultaneously, and thus better convergence speed and accuracy can be obtained. After multi-class classification and bounding box regression in the feature maps, the fast non-maximum suppression (Fast NMS) method [30] is used to retain the local maximum value of confidence score to obtain the final result.

3.2. Vehicle datasets

In general object detection, there are a number of well-known datasets. However, the datasets of PASCAL VOC Challenges [36–37] and ImageNet Large Scale Visual Recognition Challenge [38] include only a few types of vehicles such as car, bus and truck. The MS-COCO Detection Challenge dataset [39] and the KITTI dataset [40] contain images of some large vehicles, but their appearance is quite different from the appearance of the vehicles we want to detect. Therefore, the vehicle images in these datasets cannot be directly used for training the YOLO network, but instead, it is necessary to create custom vehicle datasets with less bias.

Since the detection is performed by extracting the feature map, the vehicle with similar physical characteristics can be grouped together. Therefore, based on the investigation of vehicles passing on urban bridges, the vehicle is classified into a total of nine types, which including car, sport utility vehicle (SUV), light truck, pickup truck, minibus, bus, heavy truck, tractor trailer and special vehicle. The procedure of creating datasets is as follows:

(1) To contain significant variability [36] in terms of object size, orientation, pose, illumination, position and occlusion, the images of vehicles are taken by using the camera, or by downloading from the internet. With a limited datasets, the number of images per type vehicle should not vary too much from each other in order to ensure the ability to detect different types of vehicles.

Table 1
Architecture of CSPDarknet53 (Note: BN refers to batch normalization. Mish means mish activation function. ResBlock denotes the residual block).

Type	Number	Filters	Output
Conv2D + BN + Mish	1	32	608 × 608
ResBlock	1	64	304 × 304
ResBlock	2	128	152 × 152
ResBlock	8	256	76 × 76
ResBlock	8	512	38 × 38
ResBlock	4	1024	19 × 19

(2) The random sampling method is used to divide the vehicle dataset into two parts: One part is the training dataset, accounting for 80% of the whole data, which is used for the training of the vehicle detector. The other part is the testing dataset, which accounts for 20% of the whole data and is used to evaluate the capability and accuracy of the detector.

(3) The images in datasets are annotated with two attributes: One is category tags, including car, SUV, light truck, pickup truck, minibus, bus, heavy truck, tractor trailer, and special vehicle. Another is an axis-aligned ground truth box surrounding vehicles present in the image. When there are multiple vehicles in the same image, ground truth boxes are drawn and category tags are added to different vehicles to increase the data samples. The classification and category tag of the dataset are shown in Fig. 3.

(4) The image size, category tags, and the pixel coordinates of all ground truth boxes are extracted and written to files. At the same time, each label is encoded as an integer from 0 to 8 to represent the nine different vehicle types.

To extend the training dataset, the feature adjustments and size changes for images are used. New images are formed by adjusting the brightness, contrast, saturation and hue of the image in vehicle dataset, and flipping horizontally with a 50% chance and in random order. To improve the ability to detect small targets (vehicles that occupy less space in the image), the width and height of the original image are multiplied by a randomly selected expansion ratio to calculate the size of the canvas, and then the original image is placed into that canvas to form a new image. Accordingly, by randomly cropping the target object in the original image, a new image is formed to increase the sample number of large or occluded targets.

3.3. Training of YOLO-v4

To obtain the vehicle detection model, the YOLO-v4 was trained iteratively using the created dataset. The specific training process is as follows:

(1) Select the appropriate batch size according to the computer hardware, especially the graphics processing unit (GPU). Meanwhile, in order to avoid getting into a local optimal solution, the appropriate learning rate was chosen according to the literature [34].

(2) After setting the initial parameters, the iterative training of the YOLO-v4 was started. To facilitate real-time evaluation of the detection model during training, the training results of the parameters such as weights and biases need to be saved at regular intervals.

(3) The final values of the YOLO-v4 parameters are obtained by terminating the training when the loss function is no longer decreasing, and will be used for image feature extraction and vehicle detection.

4. Vehicle 3D bounding box reconstruction

4.1. 'Bridge-Vehicle-Camera' system

To detect the vehicles on bridge decks, cameras need to be installed on both sides of the bridge deck. The camera should be in a fixed posture during operation. Meanwhile, the position and angle should be reasonable to capture 3 surfaces of the vehicles. To facilitate camera calibration, a flat area on the bridge deck is chosen as the target region for the camera. Once the camera is installed, the bridge, the vehicle and the camera form a BVCS as shown in Fig. 4. Four coordinate systems are created in the BVCS and they are world coordinate system (WCS), vehicle coordinate system (VCS), camera coordinate system (CCS) and pixel coordinate system (PCS).

(1) WCS is created on half the bridge deck. The direction parallel to the lane markings and consistent with travelling is defined as X, the direction perpendicular to the X and pointing outward from half the bridge deck is defined as Y. The Z is perpendicular to the bridge deck and

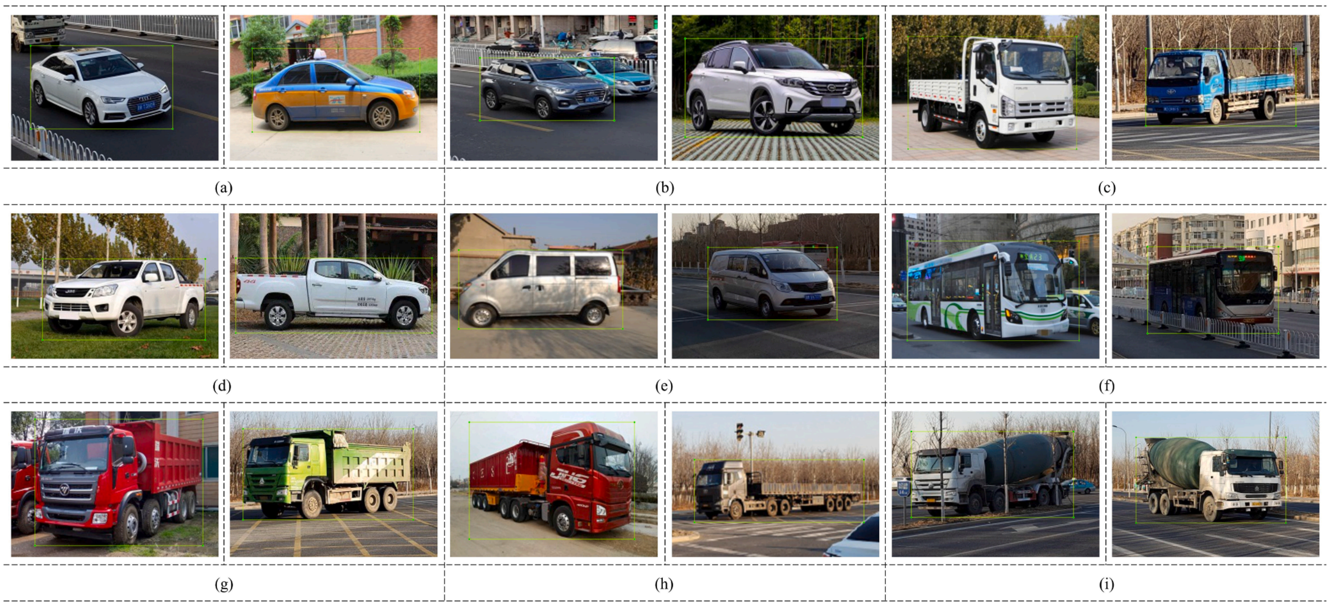

Fig. 3. Classification and annotation of the vehicles in datasets: (a) car; (b) sport utility vehicle (SUV); (c) light truck; (d) pickup truck; (e) minibus; (f) bus; (g) heavy truck; (h) tractor trailer; (i) special vehicle.

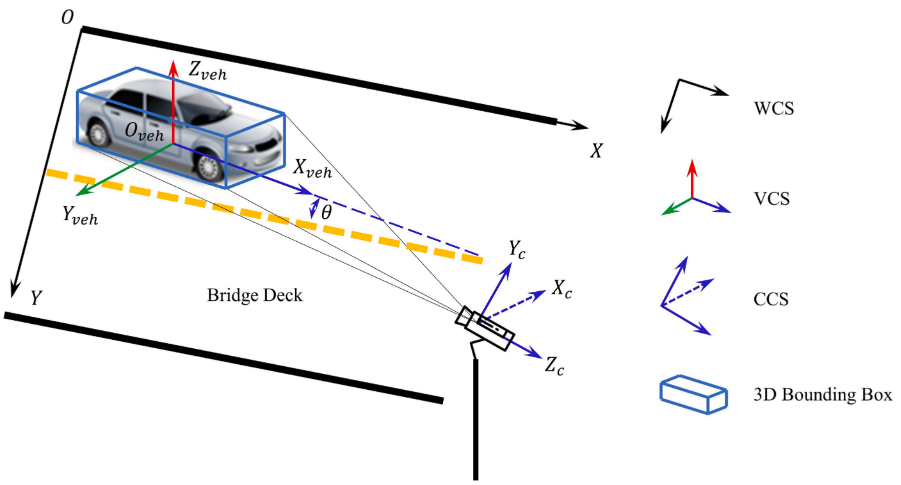

Fig. 4. 'Bridge-Vehicle-Camera' system (Note: In addition to the world coordinate system (WCS), the system also includes two local coordinate systems, vehicle coordinate system (VCS) and camera coordinate system (CCS). The 3D bounding box refers to the cuboid that wraps tightly around the body of the vehicle.)

downwards. Once the WCS is created, the lane number of the vehicle can be determined by the vehicle location coordinates, and the speed of the vehicle can be obtained by tracking the vehicle.

(2) VCS is a local coordinate system that moves with the vehicle. Each vehicle has a 3D bounding box that just encloses its body. To represent the size of the vehicle and the coordinates of the 3D bounding box vertexes, VCS is created. The origin of VCS is located at the midpoint of the bottom of 3D bounding box. The direction of travelling is defined as X_{veh}, and the direction perpendicular to X_{veh} and pointing to the outside of the bridge deck is defined as Y_{veh}. The Z_{veh} is perpendicular to the bridge deck and upwards. In most cases, X and X_{veh} are oriented in the same direction. However, when a vehicle changes lane, a certain angle between the two axes comes, which is denoted by θ.

(3) CCS is a coordinate system attached to the camera. Taking the optical center of the camera as the origin and the two axes of the image sensor as X_c and Y_c. The direction of the lens is defined as Z_c, which is shown in Fig. 4. Once the camera is fixed, the translational and rotational relationship between CCS and WCS is unique. Then the coordination of the 3D bounding box vertexes in WCS can be converted to the coordinates in CCS.

(4) PCS is created on the image plane and its origin is located at one corner of the image plane. The coordinates represent the number of rows and columns of that pixel in the image. Once the vehicle is detected, the 3D bounding box can be drawn in the image by using pixel coordinates.

Given that the coordinates of the 3D bounding box vertexes in VCS are$[X_{veh}, Y_{veh}, Z_{veh}]^T$, the coordinates of the middle point at the bottom of 3D bounding box in WCS are $\left[p_x, p_y, 0\right]^T$, the angle between X_{veh} and X is θ. For each vertex of the 3D bounding box, its coordinates can be converted from VCS to WCS using the following formula

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ 0 \end{bmatrix} + \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{vec} \\ Y_{vec} \\ Z_{vec} \end{bmatrix} \tag{2}$$

Since the positions of WCS and CCS are arbitrary, it can be assumed that there is a translation matrix $\boldsymbol{t}_{3\times1}$ and a rotation matrix $\boldsymbol{R}_{3\times3}$ between them. According to the pinhole camera model [41], the coordinate transformation from WCS to CCS can be expressed as

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\boldsymbol{K}_{3\times3}[\boldsymbol{R}_{3\times3}\quad \boldsymbol{t}_{3\times1}]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}=\boldsymbol{P}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}, \tag{3}$$

where s is an arbitrary scale factor, $\boldsymbol{P}$ is the projection transformation matrix, $[\boldsymbol{R}\ \boldsymbol{t}]$ is the camera extrinsic parameter matrix, and the camera intrinsic matrix $\boldsymbol{K}$ is given by

$$\boldsymbol{K}=\begin{bmatrix}\alpha & \gamma & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}$$

(u_0, v_0) is the coordinates of the principal point, α and β are the scale factors on the u and v axes of the image respectively, and γ is the parameter describing the skewness of the two axes.

When a camera is installed on a bridge deck, its posture relative to the bridge deck is unique. In order to obtain the matrix $\boldsymbol{P}$, the camera needs to be calibrated. Among several camera calibration methods, the representative ones are Tsai [42], Heikkila [43] and Zhang's [44] methods. However, the first two methods require accurate 3D measurement data and are sensitive to measurement errors. For Zhang's methods, the sensitivity to noise can be reduced by increasing the number of corner points on the calibration board, and no laborious measuring task is required [45]. Therefore, the flexibility provides a great convenience for the calibration of the camera on the bridge. The calibration process consists of the following two steps:

(1) Calibration in the room: Take 13 images of the small calibration board at different distances and angles, and then use them for camera calibration. The calibration results include intrinsic and extrinsic parameter matrix of the camera. For a given camera, the intrinsic parameter matrix reflects its inherent characteristics and can be saved for the secondary calibration of the camera.

(2) Calibration on the bridge deck: A large calibration board is laid flat on the middle lane of half the bridge deck with its edge next to the lane markings. After taking an image of the large calibration board and getting the pixel coordinates of each grid corner in the image, the camera is calibrated for the second time.

Once the calibration of the camera is complete, the matrix $\boldsymbol{P}$ is obtained, and the WCS is also created on the bridge deck. Calibration on the bridge deck may have some impact on traffic flow, so it is appropriate to choose a moment that the traffic flow is low, such as the early hours of the morning.

4.2. Reconstruction model

With the development of deep learning techniques, scholars have proposed various methods to perform depth estimation of 3D objects [46]. Unfortunately, these methods require labeled 3D training datasets. For a camera fixed to a bridge, such 3D data is very difficult to obtain. Therefore, inspired by these methods, we propose a 3D reconstruction model based on 2D bounding box and the statistical characteristics of vehicle sizes. In Section 3, the 2D bounding box of the vehicle can be obtained during detecting. When the 3 surfaces of the vehicle are visible in the camera lens, the 3D bounding box can be projected into image based on Eq. (2), Eq. (3), and the projection transformation matrix $\boldsymbol{P}$. After the two boxes are drawn in the same image and PCS is added, the relationship between them is shown in Fig. 5, which includes 2D bounding box, 3D bounding box, VCS ($O_{vec}X_{vec}Y_{vec}Z_{vec}$), WCS ($OXYZ$), PCS ($ouv$), and the position (u_L, u_R, v_T, v_B) of the vehicle in the image. The angle between the direction of travelling and the lane markings at a certain time is θ. The parameters vector of the vehicle is $[p_x, p_y, \theta, l, w, h]^T$, where (l, w, h) denotes half-length, half-width and height of the vehicle respectively. It can be seen from Fig. 5 that 4 points(A, B, C, D) on the 3D bounding box are on the edge of the 2D bounding box. According to section 4.1, the coordinates of the 4 points in WCS are obtained by Eq. (2) and they can be written as homogeneous coordinate$(X_*, Y_*, Z_*, 1)$, where $* = A, B, C\ or\ D$. Then Eq. (3) is used to project the 4 points into the PCS. As shown in Fig. 5, the y coordinates components of A and C after being projected are equal to the pixel coordinates in the v direction of the lower and upper edges of the 2D bounding box in the image, respectively. Similarly, the x coordinates components of B and D after being projected are equal to the pixel coordinates in the u direction of the right and the left edge of the 2D boundary box in the image, respectively. So, the constraint equations are obtained as follows

$$u_L=\left(\boldsymbol{P}\begin{bmatrix}X_D\\Y_D\\Z_D\\1\end{bmatrix}\right)_x \Big/ \left(\boldsymbol{P}\begin{bmatrix}X_D\\Y_D\\Z_D\\1\end{bmatrix}\right)_z \quad u_R=\left(\boldsymbol{P}\begin{bmatrix}X_B\\Y_B\\Z_B\\1\end{bmatrix}\right)_x \Big/ \left(\boldsymbol{P}\begin{bmatrix}X_B\\Y_B\\Z_B\\1\end{bmatrix}\right)_z$$

$$v_T=\left(\boldsymbol{P}\begin{bmatrix}X_C\\Y_C\\Z_C\\1\end{bmatrix}\right)_y \Big/ \left(\boldsymbol{P}\begin{bmatrix}X_C\\Y_C\\Z_C\\1\end{bmatrix}\right)_z \quad v_B=\left(\boldsymbol{P}\begin{bmatrix}X_A\\Y_A\\Z_A\\1\end{bmatrix}\right)_y \Big/ \left(\boldsymbol{P}\begin{bmatrix}X_A\\Y_A\\Z_A\\1\end{bmatrix}\right)_z \tag{4}$$

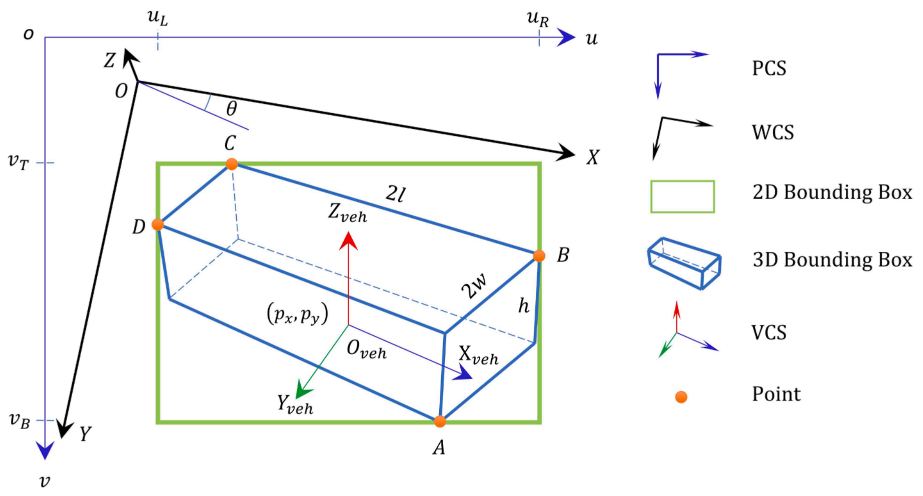

Fig. 5. Reconstruction model for vehicle 3D bounding box (Note: The 2D bounding box is the result of the vehicle detection, and 'point' is one of the vertices on the 3D bounding box. (u_L, u_R, v_T, v_B) refers to the pixel coordinates of the four edges of the 2D bounding box of the vehicle in the image. θ is the angle between the direction of travelling and the lane markings. (l, w, h) denotes half-length, half-width and height of the vehicle.)

where $()_x, ()_y$ and $()_z$ refer to the x , y and z coordinate components respectively after the projection.

When a vehicle is moving, the angle θ between the direction of travelling and the lane markings is approximately zero in most cases, except that when the vehicle is changing lanes. As a result, the parameters vector of the vehicle becomes $\left[p_x, p_y, l, w, h\right]^T$.

For convenience, the elements of the matrix $\boldsymbol{P}$ are written as

$$\boldsymbol{P} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \tag{5}$$

Combining Eq. (4) and Eq. (5) and substituting the coordinates of the 4 points (A, B, C and D) into Eq. (4), the following set of equations are obtained:

$$\begin{gathered}
\begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix}\begin{bmatrix} p_x+l \\ p_y+w \\ 0 \\ 1 \end{bmatrix} = v_B\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}\begin{bmatrix} p_x+l \\ p_y+w \\ 0 \\ 1 \end{bmatrix} \\
\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \end{bmatrix}\begin{bmatrix} p_x+l \\ p_y-w \\ -h \\ 1 \end{bmatrix} = u_R\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}\begin{bmatrix} p_x+l \\ p_y-w \\ -h \\ 1 \end{bmatrix} \\
\begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix}\begin{bmatrix} p_x-l \\ p_y-w \\ -h \\ 1 \end{bmatrix} = v_T\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}\begin{bmatrix} p_x-l \\ p_y-w \\ -h \\ 1 \end{bmatrix} \\
\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \end{bmatrix}\begin{bmatrix} p_x-l \\ p_y+w \\ -h \\ 1 \end{bmatrix} = u_L\begin{bmatrix} a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}\begin{bmatrix} p_x-l \\ p_y+w \\ -h \\ 1 \end{bmatrix}
\end{gathered} \tag{6}$$

After some arranging, a set of equations about the vehicle parameters is obtained as

$$\begin{bmatrix} a_{21}-v_Ba_{31} & a_{22}-v_Ba_{32} & a_{21}-v_Ba_{31} & a_{22}-v_Ba_{32} & 0 \\ a_{11}-u_Ra_{31} & a_{12}-u_Ra_{32} & a_{11}-u_Ra_{31} & -a_{12}+u_Ra_{32} & -a_{13}+u_Ra_{33} \\ a_{21}-v_Ta_{31} & a_{22}-v_Ta_{32} & -a_{21}+v_Ta_{31} & -a_{22}+v_Ta_{32} & -a_{23}+v_Ta_{33} \\ a_{11}-u_La_{31} & a_{12}-u_La_{32} & -a_{11}+u_La_{31} & a_{12}-u_La_{32} & -a_{13}+u_La_{33} \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ l \\ w \\ h \end{bmatrix} = \begin{bmatrix} v_Ba_{34}-a_{24} \\ u_Ra_{34}-a_{14} \\ v_Ta_{34}-a_{24} \\ u_La_{34}-a_{14} \end{bmatrix}$$

Obviously, the number of unknowns is greater than the number of equations, so the vehicle parameters cannot be obtained. In order to add constraints to that set of equations, about 3600 vehicle samples are obtained from two automotive websites in China. The statistics and analysis of vehicle sizes revealed that the length of the same type of vehicles varied considerably, but the aspect ratio r (height/width) varies very little. The same type of vehicles is picked from the overall sample, and their aspect ratios were plotted in Fig. 6. For easy to use, the aspect ratio of different types of vehicles are listed in Table 2. Based on the aspect ratio listed in Table 2, a constraint on the ratio of the height and the width is added to Eq. (7), and then the vehicle parameters can be obtained by using the following equations

$$\begin{bmatrix} a_{21}-v_Ba_{31} & a_{22}-v_Ba_{32} & a_{21}-v_Ba_{31} & a_{22}-v_Ba_{32} & 0 \\ a_{11}-u_Ra_{31} & a_{12}-u_Ra_{32} & a_{11}-u_Ra_{31} & -a_{12}+u_Ra_{32} & -a_{13}+u_Ra_{33} \\ a_{21}-v_Ta_{31} & a_{22}-v_Ta_{32} & -a_{21}+v_Ta_{31} & -a_{22}+v_Ta_{32} & -a_{23}+v_Ta_{33} \\ a_{11}-u_La_{31} & a_{12}-u_La_{32} & -a_{11}+u_La_{31} & a_{12}-u_La_{32} & -a_{13}+u_La_{33} \\ 0 & 0 & 0 & -r & 1 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ l \\ w \\ h \end{bmatrix} = \begin{bmatrix} v_Ba_{34}-a_{24} \\ u_Ra_{34}-a_{14} \\ v_Ta_{34}-a_{24} \\ u_La_{34}-a_{14} \\ 0 \end{bmatrix}$$

5. Vehicle tracking

Depending on how objects are initialized, Multiple object tracking (MOT) can be grouped into two sets: Detection-based tracking (DBT) and detection-free tracking (DFT) [47]. In section 4, the 2D bounding

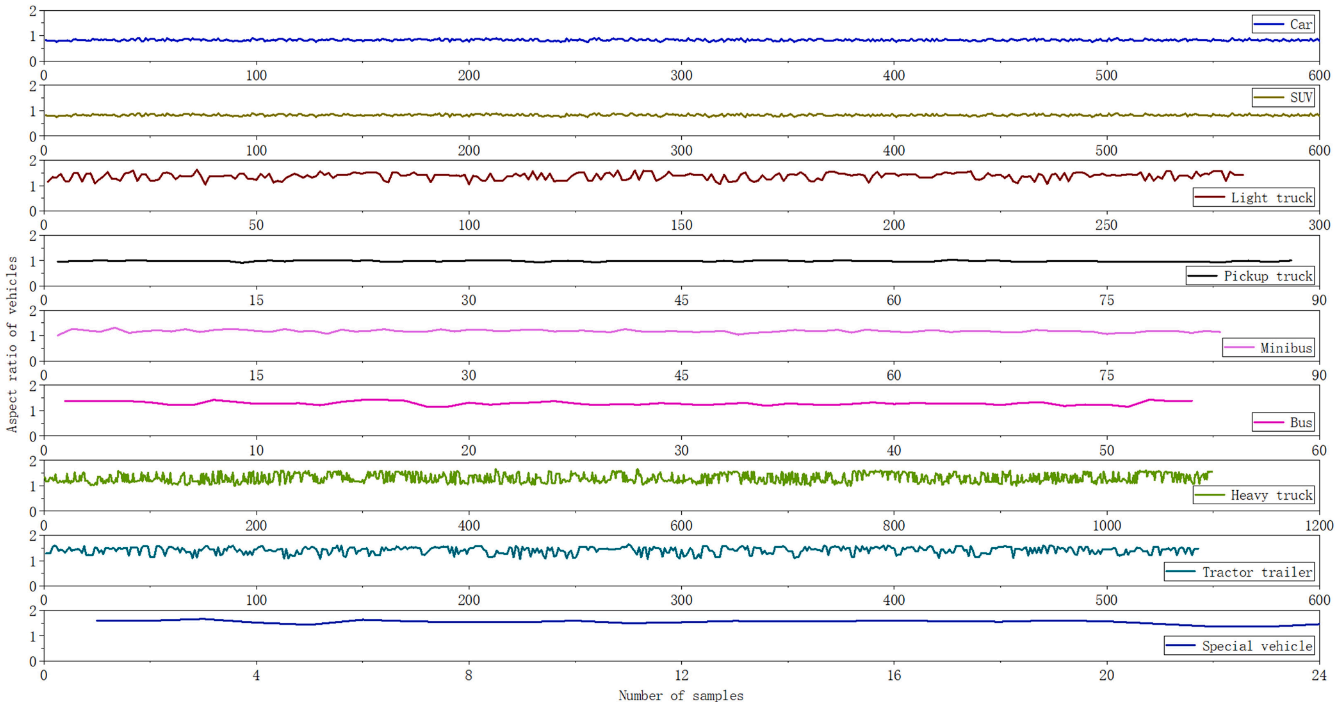

Fig. 6. Aspect ratios of different types of vehicles (Note: The horizontal axis represents the number of samples of vehicles.)

Table 2
Aspect ratios of different types of vehicles.

Type	Car	SUV	Light truck	Pickup truck	Minibus	Bus	Heavy truck	Tractor trailer	Special vehicle
Aspect ratios	0.822	0.913	1.367	0.976	1.171	1.303	1.308	1.405	1.561

box of the vehicle have been detected, so DBT is used.

From Section 4, it is clear that if the vehicle is not yet fully in the camera lens, the calculation results in incorrect information. Therefore, a ROI is created on the bridge deck. Since then, the vehicle tracking only appears in the ROI, which is shown in Fig. 7. Suppose two adjacent frames are denoted as i and j respectively and there are m vehicles in the i-th frame. In the j-th frame, l vehicles have left the ROI, while n vehicles have entered the ROI. Thus, the state matrix composed of vehicle positions in the two frames are

$$\boldsymbol{P}_i = \begin{bmatrix} V_{x_i}^1 & V_{y_i}^1 \\ V_{x_i}^2 & V_{y_i}^2 \\ \vdots & \vdots \\ V_{x_i}^m & V_{y_i}^m \end{bmatrix}, \boldsymbol{P}_j = \begin{bmatrix} V_{x_j}^{l+1} & V_{y_j}^{l+1} \\ \vdots & \vdots \\ V_{x_j}^m & V_{y_j}^m \\ \vdots & \vdots \\ V_{x_j}^{m+n} & V_{y_j}^{m+n} \end{bmatrix} (l < m)$$

The vehicle positions in the two frames are then traversed to calculate the Euclidean distance between the two vehicles as follows

$$dis_{i,j} = \sqrt{\left(V_{x_i}^M - V_{x_j}^N\right)^2 + \left(V_{y_i}^M - V_{y_j}^N\right)^2}$$
$$(M = 1, 2, \cdots, m; N = l+1, \cdots, m, \cdots, m+n), \tag{9}$$

where M and N represent the vehicle in frame i and j respectively. When the distance is less than the threshold δ, it is considered to be the same vehicle. The expression used to estimate δ is

$$0.278 \frac{v}{l_p \cdot f_r} < \delta < \frac{d}{l_p}, \tag{10}$$

where v is the maximum speed limits in km/h, l_p is the side length of the square grid in the large calibration board, its unit is in meters. f_r is the frame rate in fps. d is the minimum distance in meters between the two adjacent vehicles in the same lane. However, it is often a random number when vehicles are travelling. Therefore, δ need to be validated on-site. Although the YOLO detector has reliable detection performance, there is a probability of detection failure, which can cause interruptions in vehicle tracking. To address this problem, we predict the tracking bounding box in the current frame based on the 2D bounding box and the vehicle travel speed in the previous frame, and then use the intersection over union (IoU) between the 2D bounding box and the tracking bounding box to ensure that the vehicle is successfully tracked.

When a vehicle is being tracked, its information matrix $\boldsymbol{Minfo}_j$ including No., type, time, position coordinates (x,y), sizes (l,w,h), lane number, instantaneous speed and number of axles need to be handled with different functions as shown in Fig. 7. Of all the information, the number of axles can be obtained from Table 3. The pseudo-code of the key parts of the vehicle tracking algorithm is listed in Table 4.

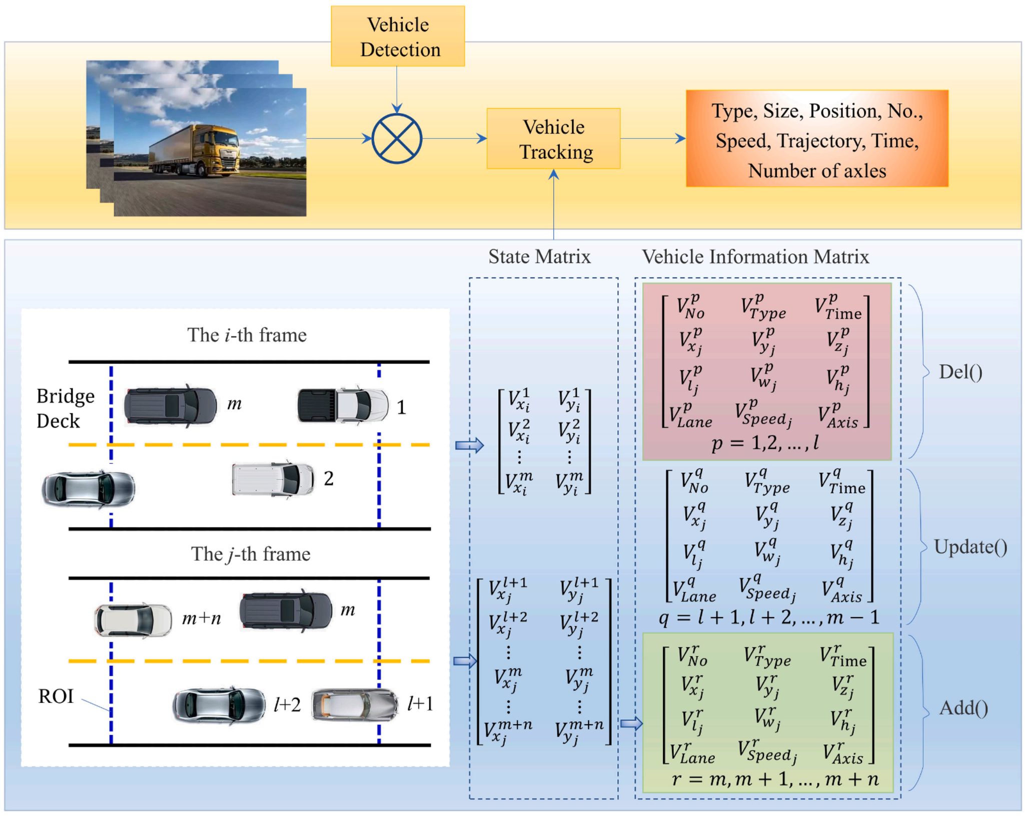

Fig. 7. Vehicle tracking algorithm (Note: Vehicle tracking is based on vehicle detection. The status and information of successfully tracked vehicles will be added or updated to the matrix, while the status and information of vehicles that drive away from the ROI will be deleted.)

Table 3
The number of axles according to the length of the heavy truck and tractor trailer [27].

The length of the vehicle (m)	≤ 9	9 ~ 10.5	10.5 ~ 13	13 ~ 14.5	≥ 14.5
The number of axles	2	3	4	5	6

$$\boldsymbol{Minfo}_j = \begin{bmatrix} V^N_{No} & V^N_{Type} & V^N_{Time} \\ V^N_{x_j} & V^N_{y_j} & V^N_{z_j} \\ V^N_{l_j} & V^N_{w_j} & V^N_{h_j} \\ V^N_{Lane} & V^N_{Speed_j} & V^N_{Axis} \end{bmatrix}$$

$(N = l+1, \cdots, m, \cdots, m+n)$

6. Application of the proposed approach

To verify the accuracy and reliability of the proposed approach, the BVLIS was developed and tested on the Silver Beach Yellow River Bridge in Lanzhou, China. As shown in Fig. 8a, it is a cable-stayed bridge with single-tower and double cable planes, which has a span of 2 × 133 m and a tower height of 79 m. Fig. 8b shows that the main bridge has a deck width of 25.5 m and 3 lanes within half the deck. The first two lanes are motorways with widths of 3 m and 3.6 m respectively, and the third lane is a mixed carriageway with a width of 2.7 m.

Table 4
The pseudocode for the vehicle tracking.

```
Algorithm : MOT for the vehicle tracking.
Input : the video of the vehicle on bridge deck, ROI, the threshold $\delta$, frame rate $f_r$;
Output : vehicle information matrix $\boldsymbol{Minfo}$;
1:  while true do
2:      $vehicleList_j \leftarrow$ Detect the vehicle frame by frame;
3:      for $k = 0$; $k <$ len($vehicleList_j$); $k$++ do
4:          $vehicle_j[k] \leftarrow vehicleList_j[k]$;
5:          $[V^k_{x_j}, V^k_{y_j}] \leftarrow$ Acquire the position of $vehicle_j[k]$;
6:          if $vehicle_j[k] \in$ ROI and $vehicleList_i$ is not null then
7:              for $l = 0$; $l <$ len($vehicleList_i$); $l$++ do
8:                  $vehicle_i[l] \leftarrow vehicleList_i[l]$;
9:                  $[V^l_{x_i}, V^l_{y_i}] \leftarrow$ Acquire the position of $vehicle_i[l]$;
10:                 $dis_{i,j} \leftarrow \sqrt{\left(V^l_{x_i} - V^k_{x_j}\right)^2 + \left(V^l_{y_i} - V^k_{y_j}\right)^2}$;
11:                 $IoU \leftarrow$ the intersection over union (IoU);
12:                 if $dis_{i,j} < \delta$ and $IoU > 0.6$
13:                     $\boldsymbol{Minfo}$[j].Update( );
14:                     $\boldsymbol{P}$[j].Update( ) $\leftarrow [V^k_{x_j}, V^k_{y_j}]$;
15:                 else
16:                     $\boldsymbol{Minfo}$[j].Add( );
17:                     $\boldsymbol{P}$[j].Add( ) $\leftarrow [V^k_{x_j}, V^k_{y_j}]$;
18:                 end if
19:             end
20:         end if
21:     end
22: end
```

6.1. Detector training and testing

Six thousand images of vehicles were taken on urban bridges and roads by using the camera under different times and locations. Of all the images, the percentages of car, SUV, light truck, pickup truck, minibus, bus, heavy truck, tractor trailer, and special vehicle are 21.85%, 15.48%, 9.18%, 8.26%, 10.26%, 14.87%, 6.84%, 6.35% and 6.91% respectively. The images were annotated with two attributes and divided into training and testing groups using a random sampling method. On the basis of the detector building and the parameters (the initial learning rate of 0.001 and a batch size of 16 images), 200,000 training sessions were performed on a computer with GPU (GeForce GTX 1080 Ti), CPU (Intel Core i7-8700K @ 3.7 GHz) and software environment (Python 3.7 and PyTorch 1.7). During the training process, the total loss function decreased sharply from the beginning to stabilize at the later stage as shown in Fig. 9a, and the precision-recall curves of nine vehicle types on testing dataset are shown in Fig. 9b. As can be seen from the Fig. 9b, the model has good detection capability. Meanwhile, the average precision (AP) for each type of vehicles was obtained and listed in Table 5.

6.2. Camera calibration

To detect the vehicle in two different directions on the bridge deck, two cameras with a resolution of 2560 × 1440 and a frame rate of 30 fps were installed on both sides of the bridge deck at the end of the main bridge as shown in Fig. 8a. The two cameras worked in the same way, so the subsequent content was for the first camera only. According to several sets of tests on the bridge deck, it was found that when the camera was located at a height of 3.8 m, all three surfaces (top, front,

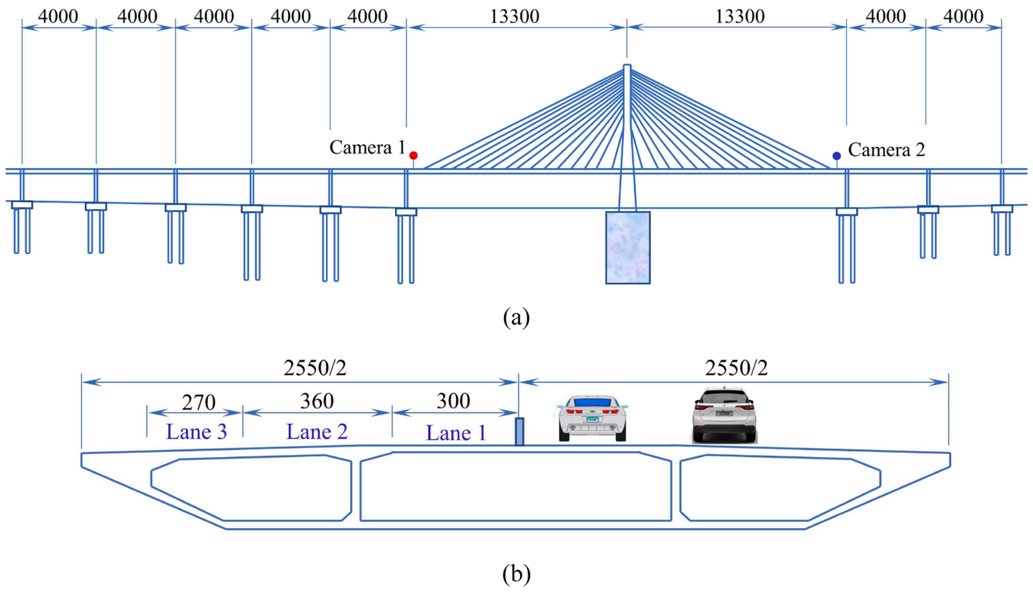

Fig. 8. The Silver Beach Yellow River Bridge (cm): (a) front view of the Silver Beach Yellow River Bridge; (b) cross-sectional view of main beam.

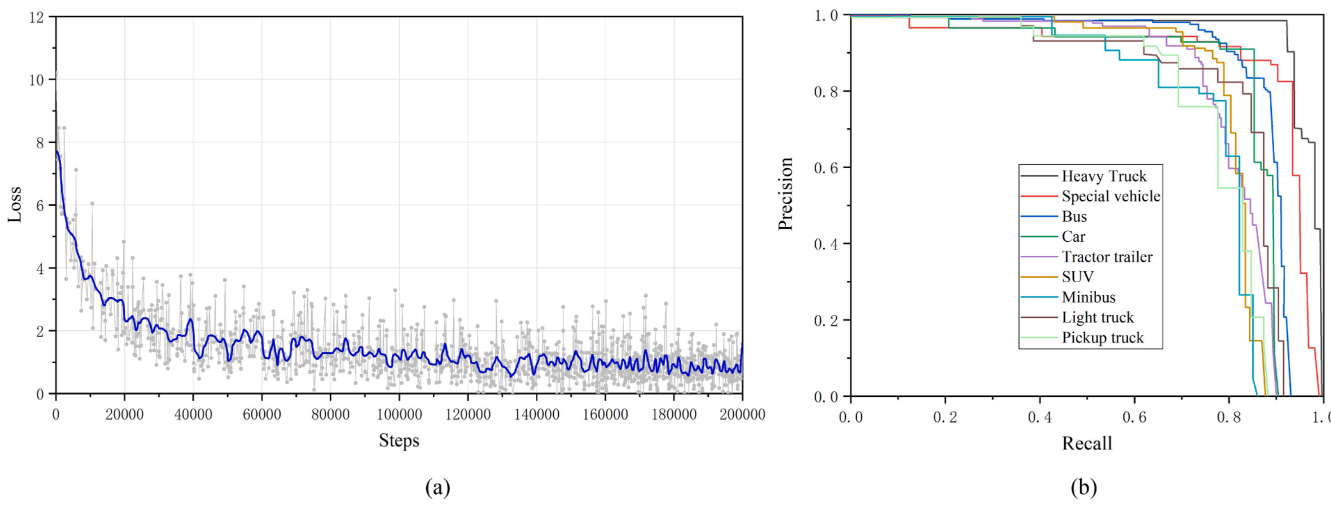

Fig. 9. Detector training and testing: (a) curve of the loss function; (b) precision-recall curves.

Table 5
APs of nine types of the vehicle on the testing dataset.

Type	Car	SUV	Light truck	Pickup truck	Minibus	Bus	Heavy truck	Tractor trailer	Special vehicle
AP (%)	89.5	82.0	77.3	76.6	81.6	90.7	96.1	85.4	92.7

and right) of the vehicle in the field of view had better visibility, thus facilitating the reconstruction of the 3D surround box.

To get the camera posture relative to the bridge deck, the calibration was required twice in total. Thirteen images of the small calibration board were taken and imported into the system, the intrinsic parameter matrix was calculated and saved as an Excel file. For checking the calibration results, the first point gathered from the board was used as the origin of a coordinate system, and the row and column on the board as X and Y of WCS respectively, the coordinate system was shown in Fig. 10a. In the second calibration, a floor leather of dimension 3×2 m with a black and white square (the side length is 27.5 mm) was selected as the large calibration board and laid on lane 2 with one side immediately adjacent to the lane markings. The bridge deck and the large calibration board were shown in Fig. 10b and 10c respectively. An image of the board was taken with the camera, the pixel coordinates of each grid corner were gathered from the image and saved as a data file. By importing the image and the data file together into the BVLIS, the projection matrix was obtained and the WCS was created on the bridge deck as shown in Fig. 10d.

To check the accuracy of the second calibration, the projection of the grid corners was drawn on the image using small yellow circles. As can be seen from Fig. 10d, there is an accurate match between the projection and the original position. The deviation of the projection of grid corners from the original position was calculated and plotted in Fig. 10e. The result shows the maximum error is no more than 1% relative to the image size.

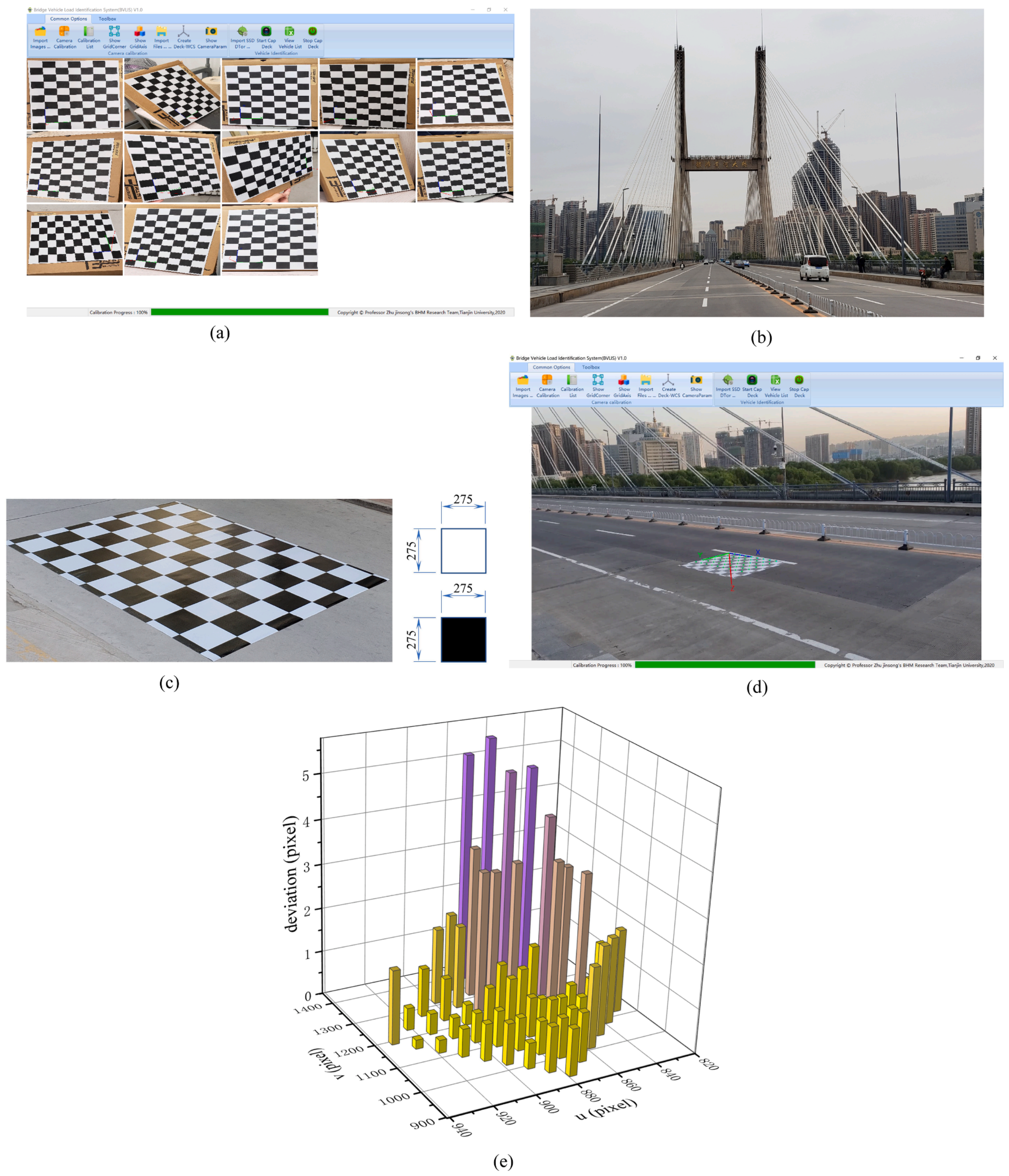

Fig. 10. Camera calibration: (a) images of the small calibration board used for the first calibration of the camera; (b) a cable-stayed bridge in operation; (c) large calibration board for the second calibration of the camera; (d) WCS was created on the bridge deck after the camera calibration; (e) the deviation in the second calibration.

6.3. *Vehicle spatiotemporal information acquisition*

Based on the camera position, location of the WCS, and the width of the lanes shown in Fig. 8b, the ROI was set as $x\in[-10,13]$ and $y\in[-15,15]$ (no unit data). The speed limit on the bridge is 60 km/h, the frame rate of the camera is 30 fps. Since the vehicle traveled at a faster speed and with larger spacing, the threshold δ was set to 3.5 in term of Eq. (10).

When the identification began, the type and confidence score of the vehicle in each frame were detected firstly, then the position coordinates in WCS and size were obtained by using the reconstruction model. The 3D bounding box and confidence score of the detected vehicle were drawn and shown in Fig. 11. For the better visibility, the 3D bounding boxes of different types of vehicles were represented in different colors. As soon as a part of the vehicle body was detected, the system calculates the position and size of the vehicle accordingly. However, when the vehicle was not fully visible in the camera lens, the calculations were unreliable.

When the vehicle was being tracked, the lane number of the vehicle was obtained by judging the relationship between the location of the vehicle and the lane markings on the bridge deck, the vehicle axle number was determined according to vehicle type and length as listed in Table 3. Once the identification and tracking of the vehicle were completed, the trajectory, instantaneous speed, average speed and the

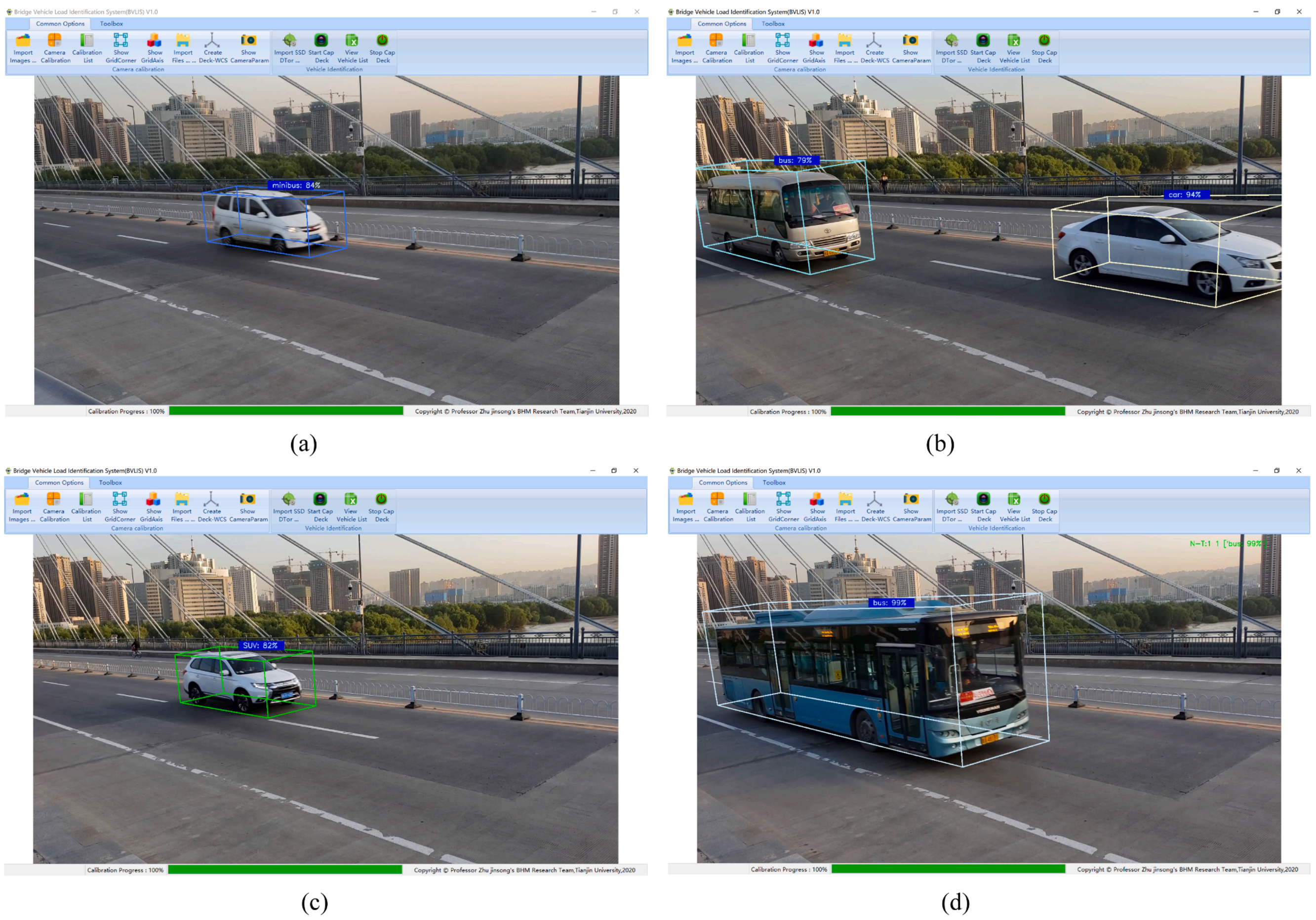
minibus: 84%
bus: 79%
car: 94%
SUV: 82%
bus: 99%
Bridge Vehicle Load Identification System(BVLIS) V1.0
Calibration Progress : 100%
Copyright © Professor Zhu Jinsong's BHM Research Team,Tianjin University,2020
(a)
(b)
(c)
(d)

Fig. 11. Reconstruction of vehicle 3D bounding box: (a) Wuling Hongguang (minibus); (b) Chevrolet Cruze (car); (c) Mitsubishi Outlander (SUV); (d) Urban bus (bus).

moment were obtained. Assuming the vehicle was traveling at the average speed over the entire bridge deck and maintaining in the same lane, the vehicle spatiotemporal information was obtained. Since the detector have a certain probability of failure, erroneous results or missed detections can cause inconsistent between the tracking data and the actual situation. Similarly, when the position of the vehicle being detected has shifted, there was an error in the calculation of the size and speed.

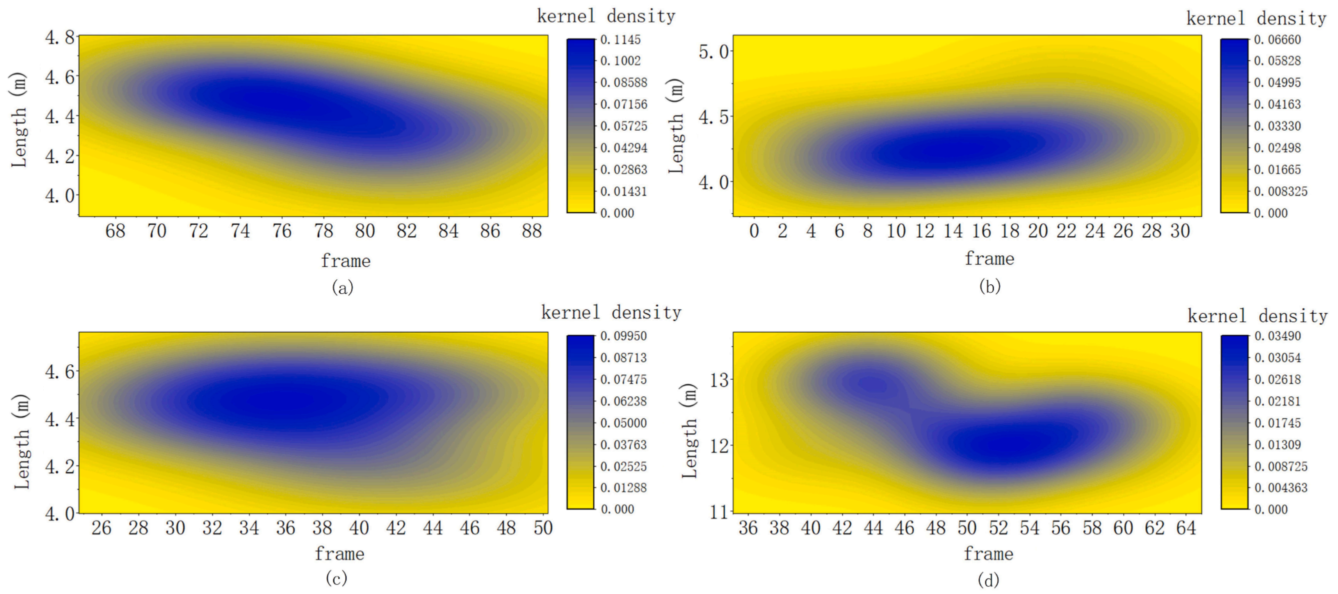

Fig. 12. Kernel density of the vehicle length: (a) Wuling Hongguang (minibus); (b) Chevrolet Cruze (car); (c) Mitsubishi Outlander (SUV); (d) Urban bus (bus). (Note: The horizontal coordinate represents the video frame number used in the calculation.)

6.4. Error analysis

After identifying and tracking the different 4 vehicles (Wuling hongguang, Mitsubishi outlander, Chevrolet cruze, and urban bus. The lengths of them are 4.5 m, 4.705 m, 4.598 m, and 11.53 m respectively.) in four videos, kernel density maps of the vehicle length were drawn as shown in Fig. 12. The diagrams show that the calculated length of the same vehicle appearing in successive frames is not fixed but approaches the actual value with great probability. When the number of detected frames reaches 10 or more, the average of the calculated length is very close to the actual size. The same is true for the width and height of the vehicle, which are not listed here due to space limitation. Fig. 13 shows the trajectory of the four vehicles. X and Y are the two axes in WCS on the bridge deck. It can be seen from the diagram that the vehicle did not change lanes while travelling, and the trajectory and the lane markings are in an approximately parallel relationship. This is consistent with the actual situation. By calculating the position of the vehicle frame by frame, the instantaneous and average speed of the four vehicles were calculated based on the frame rate of the camera as shown in Fig. 14. As can be seen from the graph, there is a wide range of variations in the instantaneous speed of the vehicle. When the continuously calculated instantaneous speed is averaged, the variation of the average speed gradually stabilizes and approaching the actual speed. In order to compare the average speed with the actual speed, a timer was used to obtain the actual speed of vehicles, which was indicated by a solid green line in Fig. 14. When the tracking is complete, the error between the average speed and the actual speed is no more than 6%.

The top five type of vehicles make up the vast majority of the total number of vehicles on urban bridges. Four videos were shot randomly for each type of vehicles in different lanes (1 and 2) and at different travelling speeds (fast and slow). After calculating the 20 sets of videos and statistically analyzing the results, the average errors in vehicle size and speed were listed in Table 6. There is some deviation of the detection result relative to the real position of the vehicle during the vehicle detection, it will bring errors to the calculation of the vehicle parameters. The error is small-scale relative to the displacement generated by the vehicle in a certain time period, but it appears larger relative to the vehicle size. Therefore, the result shows that the calculation of speed is more accurate than that of sizes.

Through calculation and analysis, it can be seen that the spatio-temporal information of moving vehicles on the bridge can be obtained in real time. However, errors in the vehicle detection, camera calibration and the constraint equation led to certain deviations between the calculated and actual results. This requires improvement of the algorithm in future research to improve the computational accuracy.

6.5. Comparison with related works

The most recent related studies are listed in Table 7. As can be seen from the table, DCNN is used for all methods of obtaining vehicle load information based on computer vision techniques. In order to calculate and extract vehicle information frame by frame, the efficiency and accuracy of the detector is critical. The latest research shows that the one-stage detector has obvious advantages, and YOLO-v4 is a significant improvement over the previous version of YOLO-v3 [35]. In our experiments, the inference time of one frame can reach 0.016 s, which fully meets the requirements of real-time detection.

In terms of algorithm implementation, there are also significant differences. In the image calibration algorithm [27], data collection was performed using standard vehicles, and then the vehicle size is obtained based on fitting and interpolation of the data. Since all vehicles are divided into a total of three granularities for calibration: large, medium and small, the results are subject to significant deviations. In addition, image calibration was required before deployment on each bridge, which undoubtedly increased the cost of the system significantly. In the approach based on reference line [30], the authors use two lines on the bridge deck as a reference to map the information in the image to 3D space, from which the position of the vehicle is obtained. However, they argue that the method is simple, but at the expense of accuracy. In a latest report [31], the authors use a dual-target detection model to achieve estimation of vehicle size and position based on body and tail detection. The method has been proved to be feasible through field testing. However, the distance between the vehicle tail and the road, as well as the difficulty of detecting the tail of the vehicle, can causes errors in calculation. In addition, the detection of the tail of the vehicle makes the annotation of the data used for detector training increase exponentially. The last two studies listed in Table 7 both try to obtain the 3D dimensions of the vehicle through spatial mapping. However, our algorithm requires only the direct solution of the system of equations without any inter-frame iterative computation.

In addition to the above, the results are analyzed and discussed in these reports. After image calibration and video calculations, the resulting vehicle speed and truck length are plotted in graphs [27]. It is estimated from the graphs that the error in the truck length is about 6% (No detailed data for reference), and approximately 6% of vehicles have an error of greater than 6% in their travel speed. In other reports

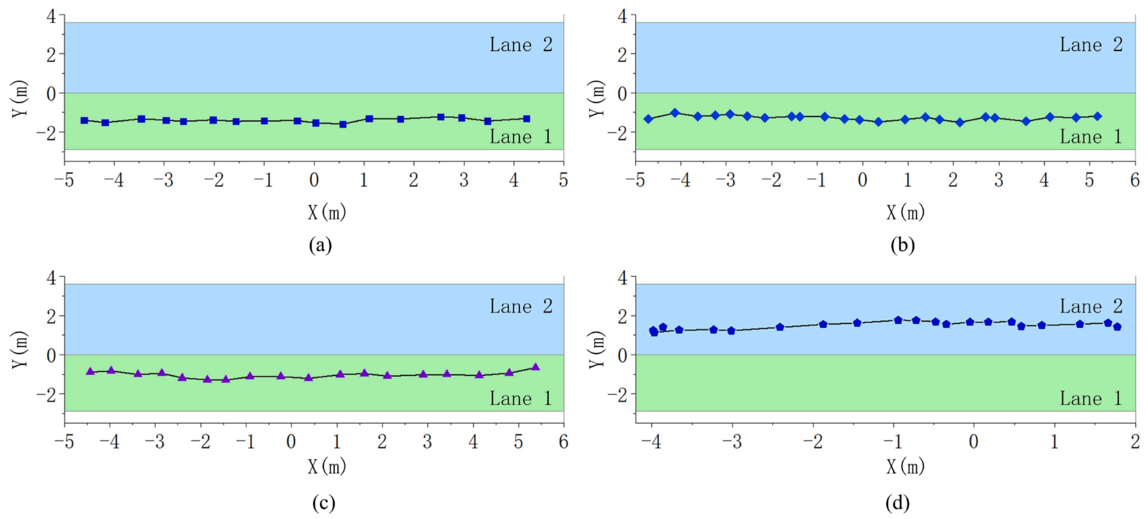

Fig. 13. Trajectory of vehicles: (a) Wuling Hongguang (minibus); (b) Chevrolet Cruze (car); (c) Mitsubishi Outlander (SUV); (d) Urban bus (bus). (Note: Taking WCS on the bridge deck as the reference coordinate system.)

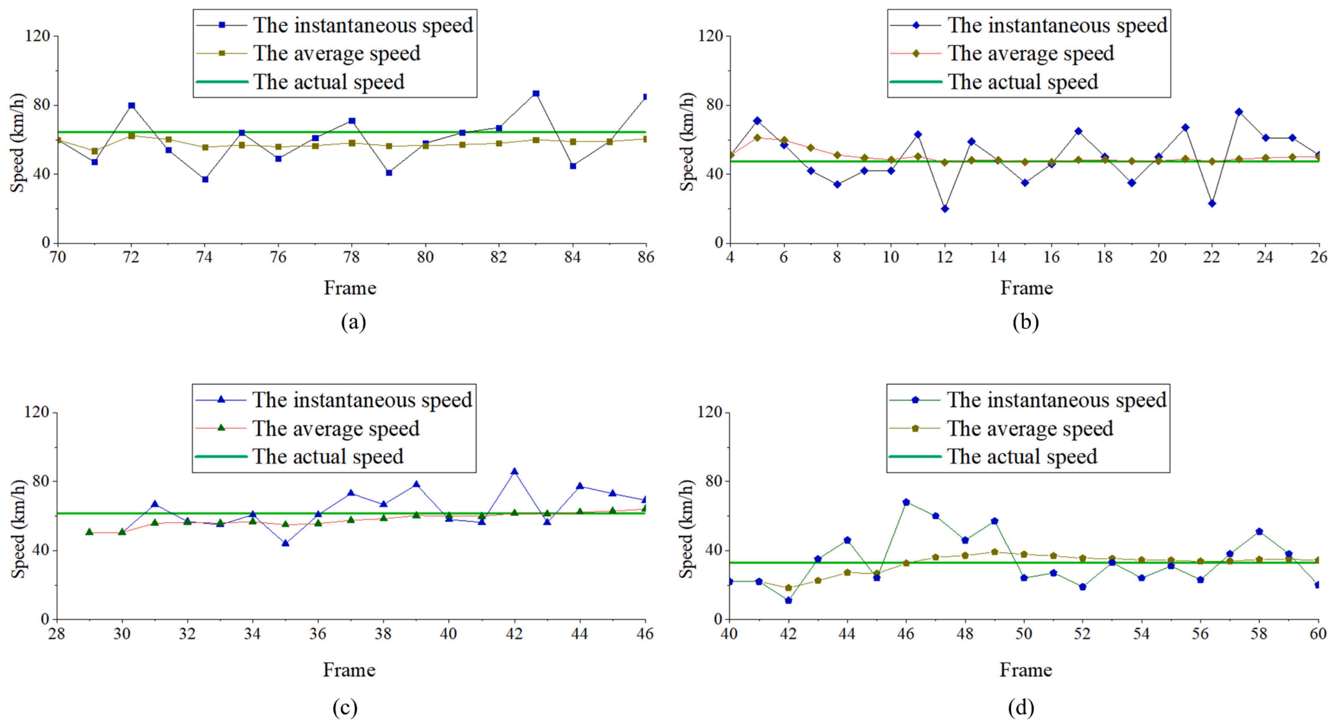

Fig. 14. Speed of vehicles: (a) Wuling Hongguang (minibus); (b) Chevrolet Cruze (car); (c) Mitsubishi Outlander (SUV); (d) Urban bus (bus). (Note: The instantaneous speed is calculated from the displacement and time generated by the vehicle between two adjacent video frames.)

Table 6
Average error in size and speed.

Type	Length (%)	Width (%)	Height (%)	Speed (%)
Minibus	2.28	8.78	3.61	5.83
SUV	6.00	2.54	3.92	2.54
Car	6.72	5.99	2.58	5.73
Bus	9.90	3.58	3.93	4.12
Light truck	3.65	4.21	7.23	3.56

[28,30], only some graphs of the calculation results are given, and no more detailed data are listed. More detailed data are given in a report [31], where the maximum average error of length and width are 18.16% and 23.09% respectively. As can be seen from the figures, the transverse position of the vehicle within 10 frames showed a large fluctuation, so the trajectory also showed a large deviation relative to the actual trajectory. As a comparison, the maximum average errors of the vehicle length and width obtained with our approach are 9.9% and 8.78%, respectively. Also, the vehicle trajectory is closer to the actual trajectory in a very short period of time.

Objectively evaluating the differences between these algorithms is challenging due to factors such as camera performance (Parameters and distortion), mounting location, working environment, and program code. However, by comparing the detector, the algorithm implementation and the existing data, our algorithm is closer to reality and therefore achieves better accuracy.

7. Conclusions

An approach for obtaining the spatiotemporal information of vehicle loads based on 3D bounding box reconstruction with computer vision is proposed. Based on the creation of vehicle datasets, training of vehicle detector, and calibration of the camera, the reconstruction model of the vehicle 3D bounding box is proposed and the algorithm equations are derived. Meanwhile, the BVLIS with GUI was developed and tested on a bridge to verify the reliability of the approach. The main conclusions are as follows:

(1) In this paper, the vehicle detection and projection matrix are obtained based on YOLO-v4 and camera calibration. Then the projection relationship between 2D and 3D bounding box is investigated and a 3D bounding box reconstruction algorithm for vehicle 3D is proposed. In contrast, our approach is closer to the actual because some assumptions and approximations that existed in the previous works are eliminated.
(2) Field tests have shown that the 3D bounding box of the vehicle can be reconstructed efficiently and accurately using the approach to obtain the size and spatial location of the vehicle. Then, based on the vehicle tracking, the spatiotemporal information of the vehicle is obtained.
(3) One linear system of equations needs to be solved to obtain the size and location of the vehicle, so the algorithm is

Table 7
Relate works.

Related works	Detector	Algorithm implementation	vehicle information			
			Size	Position	Trajectory	Speed
[27]	Faster R-CNN	Image calibration	√			√
[28]	Faster R-CNN	Detection based on DCNN		√		
[30]	Mask R-CNN	Reference line			√	
[31]	YOLO-v3	dual-target vehicle detection model	√	√	√	
Ours	YOLO-v4	3D bounding box reconstruction model	√	√	√	√

computationally compact and can be fully used for real-time identification of vehicle loads on bridge decks. Since only one calibration on the bridge deck is required, the system is easy to deploy and has little impact on traffic flow.

In the future, we will explore the identification methods of vehicle axle number and wheelbase, as well as the fusion of data from multiple cameras on the full bridge deck, to further obtain more accurate and comprehensive spatiotemporal information of vehicle loads through algorithm improvement and vehicle re-identification.

CRediT authorship contribution statement

Jinsong Zhu: Conceptualization, Methodology. **Xingtian Li:** Programming, Writing. **Chi Zhang:** Data curation, Validation. **Teng Shi:** Writing - review & editing.

Declaration of Competing Interest

The author declare that there is no conflict of interest.

Acknowledgements

This work presented here is financially supported by the National Key R&D Program of China (2018YFB1600300 and 2018YFB1600301), the National Natural Science Foundation of China (51578370) and the Tianjin Transportation Science and Technology Development Plan Project (G2018-29). Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] J. Richardson, S. Jones, A. Brown, E.J. O'Brien, D. Hajializadeh, On the use of bridge weigh-in-motion for overweight truck enforcement, Int. J. Heavy Veh. Syst. 21 (2) (2014) 83–104.
[2] J. Trott, J. Grainger, Design of a dynamic weighbridge for recording vehicle wheel loads. Road research laboratory report. Volume 219. Ministry of Tansport. London, 1968.
[3] Lee, C. E. (1966). A portable electronic scale for weighing vehicles in motion. Highway Research Record, (127).
[4] R.B. Machemehl, C.E. Lee, C.M. Walton, Acquiring traffic data by in-motion weighing, Transport. Eng. J. ASCE 101 (4) (1975) 681–689.
[5] F. Moses, Weigh-in-motion system using instrumented bridges, J. Transport. Eng.-ASCE 105 (3) (1979) 233–249.
[6] R.J. Peters, AXWAY – a system to obtain vehicle axle weights, Australian Road Res. 12 (2) (1984).
[7] B.A. Jacob, E.J. O'Brien, WAVE - a European research project on weigh-in-motion. National Traffic Data Acquisition Conference (NATDAC '96): Proceedings, 1996. Vol. 2.
[8] E.J. O'Brien, B. Jacob, European specification on vehicle weigh-in-motion of road vehicles, in: Proceedings of the 2nd European Conference on Weigh-in-Motion of Road Vehicles, 1998, 171–183.
[9] D. Cebon, Assessment of the dynamic wheel forces generated by heavy road vehicles, In: Symposium on Heavy Vehicle Suspension Characteristics, 1987, Canberra, Australia, 1988.
[10] C. O'Connor, T.H.T. Chan, Dynamic wheel loads from bridge strains, J. Struct. Eng. 114 (8) (1988) 1703–1723.
[11] T.H. Chan, S.S. Law, T.H. Yung, X.R. Yuan, An interpretive method for moving force identification, J. Sound Vib. 219 (3) (1999) 503–524.
[12] S.S. Law, T.H. Chan, Q.H. Zeng, Moving force identification: a time domain method, J. Sound Vib. 201 (1) (1997) 1–22.
[13] S.S. Law, T.H.T. Chan, Q.H. Zeng, Moving force identification - a frequency and time domains analysis, J. Dyn. Syst. Measur. Control-Trans. ASME 121 (3) (1999) 394–401.
[14] Z. Zhao, N. Uddin, E. Obrien, Field verification of a filtered measured moment strain approach to the bridge weigh-in-motion algorithm, International Conference on Weigh-in-motion (2012).
[15] L. Zhang, G. Wu, H. Li, S. Chen, Synchronous identification of damage and vehicle load on simply supported bridges based on long-gauge fiber bragg grating sensors, J. Perform. Constr. Facil 34 (1) (2020) 04019097.
[16] S.Z. Chen, G. Wu, D.C. Feng, L. Zhang, Development of a bridge weigh-in-motion system based on long-gauge fiber bragg grating sensors, J. Bridge Eng. 23 (9) (2018) 04018063.
[17] C.P. Chou, C.Y. Wang, Identification of equivalent traffic load on bridge using optical fiber strain sensors, Int. Conf. Heavy Vehicles HVParis 2008 (2013) 475–484.
[18] M. Lydon, S.E. Taylor, D. Robinson, P. Callender, C. Doherty, S.K. Grattan, E. J. O'Brien, Development of a bridge weigh-in-motion sensor: performance comparison using fiber optic and electric resistance strain sensor systems, IEEE Sens. J. 14 (12) (2014) 4284–4296.
[19] T. Khuc, T.A. Nguyen, H. Dao, F.N. Catbas, Swaying displacement measurement for structural monitoring using computer vision and an unmanned aerial vehicle, Measurement 159 (2020), 107769.
[20] X.W. Ye, T. Jin, C.B. Yun, A review on deep learning-based structural health monitoring of civil infrastructures, Smart Struct. Syst. 24 (5) (2019) 567–585.
[21] G. Chen, Q. Liang, W. Zhong, X. Gao, F. Cui, Homography-based measurement of bridge vibration using UAV and DIC method, Measurement 170 (2021), 108683.
[22] D. Dan, Q. Dan, Automatic recognition of surface cracks in bridges based on 2D-APES and mobile machine vision, Measurement 168 (2021), 108429.
[23] T. Ojio, C.H. Carey, E.J. OBrien, C. Doherty, S.E. Taylor, Contactless bridge weigh-in-motion, J. Bridge Eng. 21 (7) (2016) 04016032.
[24] M.Q. Feng, R.Y. Leung, C.M. Eckersley, Non-contact vehicle weigh-in-motion using computer vision, Measurement 153 (2020), 107415.
[25] Z. Chen, H. Li, Y. Bao, N. Li, Y. Jin, Identification of spatio-temporal distribution of vehicle loads on long-span bridges using computer vision technology, Struct. Control Health Monit. 23 (3) (2016) 517–534.
[26] D. Dan, L. Ge, X. Yan, Identification of moving loads based on the information fusion of weigh-in-motion system and multiple camera machine vision, Measurement 144 (2019) 155–166.
[27] B. Zhang, L. Zhou, J. Zhang, A methodology for obtaining spatiotemporal information of the vehicles on bridges based on computer vision, Comput.-Aided Civ. Infrastruct. Eng. 34 (6) (2019) 471–487.
[28] Y. Zhou, Y. Pei, Z. Li, L. Fang, Y. Zhao, W. Yi, Vehicle weight identification system for spatiotemporal load distribution on bridges based on non-contact machine vision technology and deep learning algorithms, Measurement 159 (2020), 107801.
[29] X. Jian, Y. Xia, J.A. Lozano-Galant, L. Sun, Traffic sensing methodology combining influence line theory and computer vision techniques for girder bridges, J. Sens. 2019 (2019) 1–15.
[30] Y. Xia, X. Jian, B. Yan, D. Su, Infrastructure safety oriented traffic load monitoring using multi-sensor and single camera for short and medium span bridges, Remote Sens. 11 (22) (2019) 2651.
[31] L. Ge, D. Dan, H. Li, An accurate and robust monitoring method of full-bridge traffic load distribution based on YOLO-v3 machine vision, Struct. Control Health Monit. 27 (12) (2020).
[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inform. Process. Syst. 25 (2012) 1097–1105.
[33] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, M. Pietikäinen, Deep learning for generic object detection: a survey, Int. J. Comput. Vision 128 (2) (2020) 261–318.
[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, SSD: single shot multibox detector. In European Conference on Computer Vision. Springer, Cham, 2016, pp. 21–37.
[35] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: optimal speed and accuracy of object detection, ArXiv Preprint (2020). ArXiv:2004.10934.
[36] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vision 88 (2) (2010) 303–338.
[37] M. Everingham, S.M. Eslami, L. Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vision 111 (1) (2015) 98–136.
[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, M. Bernstein, ImageNet large scale visual recognition challenge, Int. J. Comput. Vision 115 (3) (2015) 211–252.
[39] T.-Y. Lin, M. Maire, S.J. Belongie, J. Hays, P. Perona, D. Ramanan, Microsoft COCO: Common objects in context. In European Conference on Computer Vision, 2014, pp. 740–755.
[40] A. Geiger, P. Lenz, R. Urtasun, We ready for autonomous driving? The KITTI vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, p. 3354–3361.
[41] Z. Zhang, Camera calibration with one-dimensional objects, IEEE Trans. Pattern Anal. Mach. Intell. 26 (7) (2004) 892–899.
[42] R.Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, Radiometry (1992) 221–244.
[43] J. Heikkila, Geometric camera calibration using circular control points, IEEE Trans. Pattern Anal. Mach. Intell. 22 (10) (2000) 1066–1077.
[44] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1330–1334.
[45] W. Sun, R. Cooperstock, An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques, Mach. Vision Appl. 17 (1) (2006) 51–67.
[46] E. Arnold, O.Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, A. Mouzakitis, A survey on 3D object detection methods for autonomous driving applications, IEEE Trans. Intell. Transp. Syst. 20 (10) (2019) 3782–3795.
[47] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, T.-K. Kim, Multiple object tracking: a literature review, Artif. Intell. 293 (2021), 103448.