High-Resolution Vehicle Trajectory Extraction and Denoising From Aerial Videos

Xinqiang Chen, *Member, IEEE*, Zhibin Li, Yongsheng Yang, *Member, IEEE*, Lei Qi, and Ruimin Ke, *Member, IEEE*

***Abstract*—In recent years, unmanned aerial vehicle (UAV) has become an increasingly popular tool for traffic monitoring and data collection on highways due to its advantage of low cost, high resolution, good flexibility, and wide spatial coverage. Extracting high-resolution vehicle trajectory data from aerial videos taken by a UAV flying over target highway segment becomes a critical research task for traffic flow modeling and analysis. This study aims at proposing a novel methodological framework for automatic and accurate vehicle trajectory extraction from aerial videos. The method starts by developing an ensemble detector to detect vehicles in the target region. Then, the kernelized correlation filter is applied to track vehicles fast and accurately. After that, a mapping algorithm is proposed to transform vehicle positions from the Cartesian coordinates in image to the Frenet coordinates to extract raw vehicle trajectories along the roadway curves. The data denoising is then performed using a wavelet transform to eliminate the biased vehicle trajectory positions. Our method is tested on two aerial videos taken on different urban expressway segments in both peak and non-peak hours on weekdays. The extracted vehicle trajectories are compared with manual calibrated data to testify the framework performance. The experimental results show that the proposed method successfully extracts vehicle trajectories with a high accuracy: the measurement error of Mean Squared Deviation is 2.301 m, the Root-mean-square deviation is 0.175 m, and the Pearson correlation coefficient is 0.999. The video and trajectory data in this study are publicly accessible for serving as benchmark at *https://seutraffic.com*.**

***Index Terms*—Vehicle trajectory, unmanned aerial vehicle, vehicle detection, vehicle tracking, data quality control.**

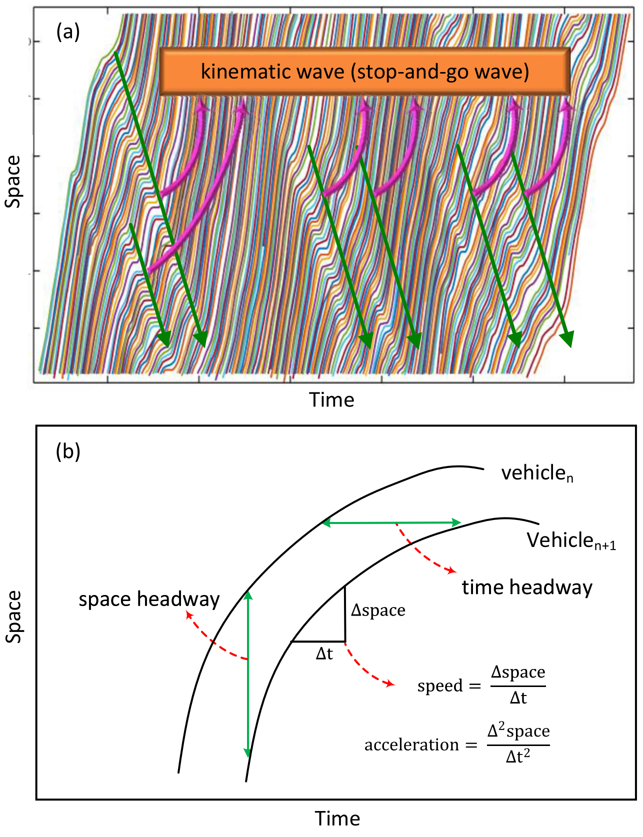


Fig. 1. Time-space map of vehicle trajectories: (a) kinematic waves in trajectories; (b) microscopic parameters in trajectories. (NGSIM, US-101, lane 1, 8:25-8:35 am).

I. Introduction

HIGH-RESOLUTION vehicle trajectory data contains rich and critical information for traffic flow studies. A trajectory map (see Fig. 1) not only supports extracting macroscopic traffic parameters such as average speed, flow and density which are the outputs from inductive loop detectors, but also indicates microscopic vehicle driving information such as vehicle speed, acceleration/deceleration, headway, gap distance, etc. Previously, numerous studies have utilized vehicle trajectory data for the purposes of traffic flow model calibration [1]–[3], traffic feature/phenomena exploration [4], [5], car-following and lane-changing behavior analysis [6], [7], and driving strategy development [8]–[11].

The most widely used trajectory data was published in the Next Generation Simulation (NGSIM) database by the U.S. Federal Highway Administration in 2006 [12]. The NGSIM data was originally collected by extracting trajectory of each

Manuscript received April 3, 2019; revised December 29, 2019 and May 3, 2020; accepted May 13, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 71871057, Grant 51579143, and Grant 51709167, and in part by the Shanghai Committee of Science and Technology in China under Grant 18040501700, Grant 18295801100, and Grant 17595810300. The Associate Editor for this article was H. Huang. *(Corresponding author: Zhibin Li.)*

Xinqiang Chen and Yongsheng Yang are with the Institute of Logistics Science and Engineering, Shanghai Maritime University, Shanghai 201306, China (e-mail: chenxinqiang@stu.shmtu.edu.cn; yangys@shmtu.edu.cn).

Zhibin Li is with the School of Transportation, Southeast University, Nanjing 210096, China (e-mail: lizhibin@seu.edu.cn).

Lei Qi is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: qilei.cs@gmail.com).

Ruimin Ke is with the Department of Civil and Environmental Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: ker27@uw.edu).

Digital Object Identifier 10.1109/TITS.2020.3003782

vehicle in multiple videos, which were shot by cameras installed on nearby building roofs. The sampling frequency of the NGSIM trajectory is 0.1 second, and each sample includes information such as instantaneous speed, acceleration/deceleration, longitudinal and lateral positions, vehicle length, vehicle type, lane ID, etc. The trajectory data are gathered on two highway segments on US-101 and I-80, and two arterial segments on Lankershim and Peachtree.

NGSIM data has motivated and supported tremendous theoretical and empirical traffic flow studies. However, the dataset contains the following limitations [13], [14]: 1) traffic states contained in the NGSIM dataset are limited to congested conditions, and the time-space scope is very limited; 2) the trajectory collection method requires numerous manual operations (i.e. manually add detect-missing vehicles into the detection results and remove detection outliers) to gain high-fidelity data; and 3) the original NGSIM videos were stitched from several highly-synchronized cameras recorded videos, resulting in calculation procedure for vehicle moving distance contains cumulative image registration errors and outliers [14], [15]. Thus, the trajectory extraction method used in the NGSIM is not friendly-transferrable. The dataset cannot support investigations of traffic flow and driver behaviors at other traffic states or state transition periods at longer time duration and larger space scopes. There are strong needs of developing accurate vehicle trajectory extraction methods to expand vehicle trajectory data in more environments.

The increasing popularity of unmanned aerial vehicle (UAV) provides a new opportunity of obtaining vehicle trajectories. Specifically, we can fly a UAV carrying high-resolution camera over target areas with shooting coverage over several hundred meters, on the basis of obeying local regulations, and capturing videos at interested time spans. The UAV-based method has advantages such as low cost, high flight altitude, wide camera coverage range, and stable flight posture [16]. In addition, the newly developed power-tethered UAV is able to stay in sky with much longer time (possibly in hours) for data collection, as the tether cable can sustainably supply power to the aircraft. The surveillance range of power-tethered UAV could be limited. The UAV flying time is expected to be longer due to battery technology promotion. For instance, the newly emerged hydrogen powered UAV can hover around the monitoring area for at least 2 hours. In that manner, we believe that UAV videos can support a large amount of real-world traffic applications such as vehicle trajectory extraction, driving behavior analysis, etc.

Extracting high-fidelity vehicle trajectory from aerial videos is a challenging task. Some studies applied the detection-based methods which try to detect vehicles through features of vehicle edges and contours. For example, Azevedo *et al.* used a scale-invariant feature transform algorithm for vehicle detection, and then applied a motion-based optimization algorithm for extracting vehicle trajectories [17]. Our previous studies proposed the optical flow-based framework to detect vehicles in UAV videos, then trained an ensemble classifier to track vehicles in consecutive image frames [18], [19]. The main challenge is how to match detected vehicles in neighboring frames, which is usually very sensitive to the successfulness of vehicle detection. In other words, if a vehicle is lost in a few frames, the algorithm may mismatch the vehicle with others and generate wrong trajectories that are unable to be rectified.

Some other studies applied methods for tracking target vehicle positions in consecutive video frames by exploring the maximum similarity between candidate vehicles and input training samples. For example, Coifman *et al.* developed a feature-based vehicle tracking framework for estimating trajectories from traffic monitoring videos [20]. Guido *et al.* applied the Haar classifier and Gaussian-blurring filters for vehicle trajectory tracking [21]. Similar research can also be found in [22], [23]. However, current methods are infeasible for large-scale vehicle trajectory extraction. In addition, existing trajectory extraction methods generally lack data quality control procedures so that the obtained trajectories may contain unexpected outliers.

Recently, the emerging computer vision technologies show great potentials in the object detection and tracking [24]–[29]. However, it is still unclear how to integrate these methods to achieve the objective of automatic and accurate vehicle trajectory extraction. In this study, we aim at filling this gap by proposing a methodological framework for obtaining vehicle trajectories from aerial videos. We compared the data outputs of our method with the manually calibrated trajectories for validation purpose. The study can support enriching trajectory dataset in more environments for further traffic flow studies.

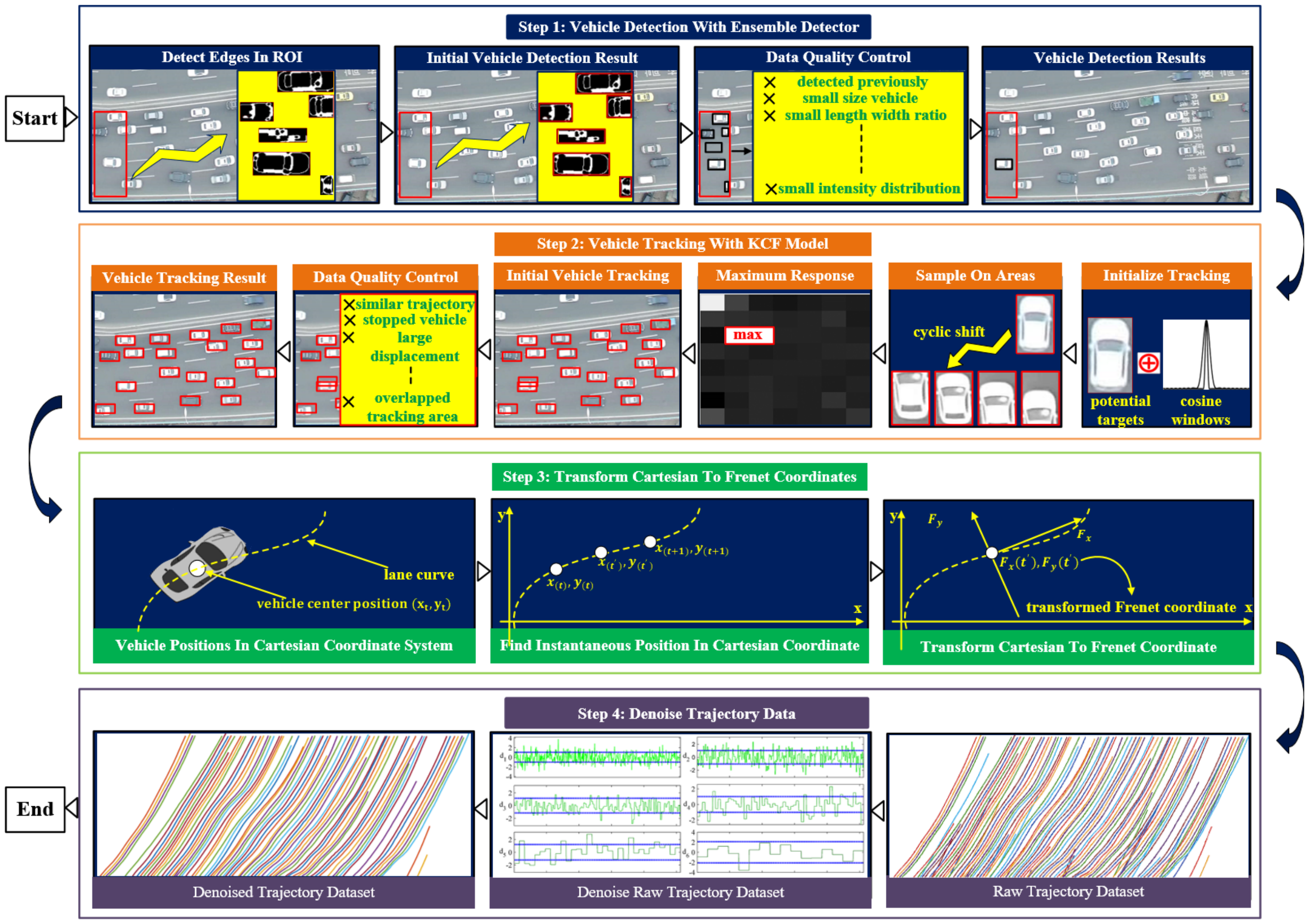
II. Methodology

A. Overall Framework

The framework proposed for vehicle trajectory extraction includes four steps (as shown in Fig. 2), which are the vehicle detection, vehicle tracking, coordinate transformation, and trajectory denoising. The purpose of the first step is to automatically detect vehicles in the region of interest (ROI) on each lane in consecutive frames using a Canny-based ensemble detector. After obtaining vehicle positions, the second step implements a fast and accurate vehicle tracking algorithm using the kernelized correlation filter (KCF). In the third step, we map the tracked vehicle positions from the Cartesian coordinate in the video to the Frenent coordinate along the road curves. A data quality control procedure runs throughout the above three steps to eliminate errors in vehicle detection, tracking, and position mapping. A general denoising algorithm of Wavelet Transformation (WT) is performed in the end, to remove position outliers and abnormal oscillations in the raw vehicle trajectories. Details of the algorithms are given in the following sections.

B. Vehicle Detection With Ensemble Detector

Though regular traffic surveillance camera can obtain high resolution video clips (vehicles contours are easily identified in images), the traffic cameras can only cover limited monitoring area which may reduce its usage in traffic applications. Vehicles are more likely to be occluded by neighboring vehicles in the videos shot at tilting angles, and vehicles far away from the camera are difficult to be accurately recognized. The above two factors significantly reduce the vehicle trajectory extraction.



Start
Step 1: Vehicle Detection With Ensemble Detector
Detect Edges In ROI
Initial Vehicle Detection Result
Data Quality Control
detected previously
small size vehicle
small length width ratio
small intensity distribution
Vehicle Detection Results
Step 2: Vehicle Tracking With KCF Model
Vehicle Tracking Result
Data Quality Control
similar trajectory
stopped vehicle
large displacement
overlapped tracking area
Initial Vehicle Tracking
Maximum Response
max
Sample On Areas
cyclic shift
Initialize Tracking
potential targets
cosine windows
Step 3: Transform Cartesian To Frenet Coordinates
lane curve
vehicle center position (x_t, y_t)
Vehicle Positions In Cartesian Coordinate System
$x_{(t)}, y_{(t)}$
$x_{(t')}, y_{(t')}$
$x_{(t+1)}, y_{(t+1)}$
Find Instantaneous Position In Cartesian Coordinate
F_y
F_x
$F_x(t'), F_y(t')$
transformed Frenet coordinate
Transform Cartesian To Frenet Coordinate
Step 4: Denoise Trajectory Data
Denoised Trajectory Dataset
Denoise Raw Trajectory Dataset
Raw Trajectory Dataset
End

Fig. 2. Schematic diagram of the proposed framework.

The UAV camera can cover a much larger monitoring area while vehicles are not occluded by other vehicles due to the top-view recording angle. The challenge is that vehicle contours from aerial videos are small and vehicle intensities are close to road intensities. In our study, we employ an ensemble Canny based edge detector to detect vehicles in the ROI at the beginning of the road segment. The main advantages of the developed detector lie in the fact that it makes use of vehicle imaging information to detect the horizontal, vertical and diagonal edges, and thus can produce a robust and accurate vehicle edge detection result. The proposed vehicle detection procedure includes three stages. In the first stage, all possible edges in the ROI are detected by the Canny edge detector, and the edges are then smoothed by the Morphology close operation. In the second stage, we merge the edges into rectangles (i.e. a rectangle is a detected vehicle) according to the connectivity criterion. In the third stage, the ensemble detector removes obvious detection outliers via data quality control.

For a given UAV frame, the Canny edge detector determines vehicle edge gradient $\mathrm{G}(\mathrm{i},\mathrm{j})$ and direction $\theta(\mathrm{i},\mathrm{j})$ by Eq. (1) and (2) [30]. Due to unevenly edge gradient and intensity distribution, the Canny detector considers some edge pixels (with gradient/intensity lower than threshold) as background and suppress them from vehicle edges. Thus, it is observed that small holes break certain edges into several separated edges, and edges from same vehicle may also be disconnected from neighbors. Morphology close operator can remove such small holes in edges, and merge separated edges into connected edges without introducing new outliers.

The Morphology close operator (see Eq. (3)) is employed to reconstruct vehicle contours.

$$\mathrm{G}(\mathrm{i},\mathrm{j}) = \sqrt{\mathrm{P}^2(\mathrm{i},\mathrm{j}) + \mathrm{Q}^2(\mathrm{i},\mathrm{j})} \tag{1}$$

$$\theta(\mathrm{i},\mathrm{j}) = \arctan\frac{\mathrm{Q}(\mathrm{i},\mathrm{j})}{\mathrm{P}(\mathrm{i},\mathrm{j})} \tag{2}$$

$$\mathrm{I}'_{\mathrm{edge}} \cdot S = \left(\mathrm{I}_{\mathrm{edge}} \oplus \mathrm{S}\right) \ominus \mathrm{S} \tag{3}$$

where i and j denote x- and y- coordinates of pixels. Symbol $\mathrm{I}_{\mathrm{edge}}$ is the edge collection detected by the Canny edge detector, $\mathrm{I}'_{\mathrm{edge}}$ is the connected edges collection, and S is structure element for filling holes in vehicle edges. Symbol $\cdot$ is the Morphology close operation, and operators $\oplus$ and $\ominus$ are Morphology dilation and erosion operations, respectively. More details about the Morphology close operator can be found in [31].

After detecting and connecting vehicle edges, we then employ the eight-connectivity criterion to determine the number of connected regions (i.e., detected vehicles), where each connected region is represented by a maximum external bounding rectangle. The number of connected regions is the number of detected vehicles in the ROI in current frame.

A careful examination on the vehicle detection results shows several typical outliers: 1) no-vehicle in the detected rectangle; 2) small-square rectangles; 3) abnormal length-width ratio; and 4) duplicated detected vehicles in neighboring frames. More specifically, the detected vehicles are considered as outliers and removed from the detection results if they satisfy one or more of the following constraints:

- The intensity distribution range of the detected vehicle is smaller than the threshold γ. This is because ROI edge pixels (neighboring to lane separators) have higher intensity than background pixels in the ROI, but lower than vehicle pixels intensity. The edges are likely to be wrongly detected as vehicle edges if no vehicle enters in the ROI;
- The ratio between the detected-vehicle square and detection ROI is smaller than the threshold ξ, because part of vehicles (such as trunk, bumper) can be detected as a vehicle if vehicle is in the border of the ROI;
- The vehicle's length-width ratio (or width-length ratio) is smaller than the threshold δ, because some roadway pixels may also be detected as vehicles as roadway pixel intensity is close to vehicles;
- Two detected vehicles in frames F_i and F_j overlap with each other and the overlap ratio is larger than α; meanwhile the frame difference is smaller than the threshold N_f, because the two vehicles are actually the same one. In such case, we merge the two detected rectangles into a larger rectangle, and the raw detection results are updated by the new detection result.

Note that emerging deep learning models can also be employed in the vehicle detection task. The main challenge is that it is difficult to connect vehicle trajectories in neighboring frames when vehicles are miss-detected by the deep learning models (i.e., vehicle temporal-spatial information may be wrongly matched due to vehicle miss-detection). Our proposed vehicle detection model obtains high accuracy (very close to the deep learning model), and the vehicle tracking model is more accurate and robust for obtaining vehicle trajectory information from image frames.

C. Vehicle Tracking With KCF Algorithm

The main challenge of vehicle tracking in UAV frames is that vehicles may be sheltered by obstacles such as trees, overhead gantries, and lights, resulting in that those obstacles are likely to be tracked as vehicles. The advantage of the KCF algorithm applied in our framework is that it trains the vehicle tracker with any possible shifts (both vertically and horizontally) of a base vehicle sample, and determines the maximum response between the tracker and the candidates. In such way, vehicle pixels in the ROI is more likely to generate the maximum response, and the obstacle-shelter interference can be suppressed.

Given a set of training vehicle patterns and labels (x_i, y_i), …, (x_m, y_m) from a UAV video, we can train a vehicle tracker V(x) by finding parameter setups that can minimize the regularization risk during vehicle tracking procedure. More specifically, the vehicle tracker is presented in the form of $V(x) = \langle t, x\rangle + b$ (symbol $\langle \cdot, \cdot \rangle$ is dot product operator), and the minimization problem for training a vehicle tracker is to find optimal solution for the following formula:

$$\min_{w,b} \sum_i L(y_i, V(x_i)) + \lambda ||t||^2 \tag{4}$$

where symbol L(y, V(x)) is a loss function, while λ controls the amount of regularization for the vehicle tracker.

To find solution for Eq. (4), we first map the input training vehicle images (i.e., training vehicle samples) to feature space $\varphi(x)$, which is defined by the kernel $g\left(x,x'\right) = \langle \varphi(x), \varphi(x')\rangle$, and solution for Eq. (4) is actually a linear combination of the inputs: $t = \sum_i \alpha_i \varphi(x_i)$ [32]. We can find a closed-form solution for Eq. (4) as follows:

$$\alpha = (G + \lambda I)^{-1} y \tag{5}$$

where G is a kernel matrix with elements in the form of $G_{ij} = g(x_i, x_j)$, parameter I is the identity matrix, and y is a group of y_i. The solution t for the vehicle tracker is implicitly represented by the vector α, where each element of α is the coefficient α_i.

To implement efficient vehicle tracking, an $n \times n$ circulant matrix $C\left(p'\right)$ is obtained from the $n \times 1$ vector p' by concatenating all possible cyclic shifts of matrix p'. Since the product $C\left(p'\right) q$ is convolution of vectors p' and q [33], we can find the solution for the product in the Fourier domain as follows:

$$C\left(p'\right) q = F^{-1}(F^*(p') \odot F(q)) \tag{6}$$

where the symbol $\odot$ is an element-wise product operation, parameters F and F^{-1} denote the Fourier and the inverse transformation, respectively, symbol $*$ is complex-conjugate operation.

Given a single UAV frame x (denoted as a $n \times 1$ vector), the dense vehicle training samples are defined as x_i (see Eq. (7)):

$$x_i = P^i x \qquad \forall i = 0, \ldots, n-1 \tag{7}$$

where parameter P is a permutation matrix that cyclically shifts the vehicle sample vectors by one element each time. The dense vehicle samples x_i are all possible translated versions of the single UAV frame.

To find an optimum solution for Eq. (5), we also define the vector g with elements as follows:

$$g_i = g(x, P^i) \qquad \forall i = 0, \ldots, n-1 \tag{8}$$

where g_i is a highly coupled representation of the kernel matrix $G = C(g)$.

We can implement operations (including multiplication and inversion) on matrices with form $C(p')$ in the element-wise manner on vectors p', on the premise of that all operations, tracking and training samples can be transformed to Fourier domain. We can obtain optimal solution for Eq. (5) as follows:

$$\alpha = F^{-1}\left(\frac{F(y)}{F(k) + \lambda}\right) \tag{9}$$

where division operation is implemented in the element-wise manner.

Based on the results of Eq. (8) and (9), the KCF algorithm finds target vehicles in each UAV frame easily. Specifically, the KCF algorithm reckons the image region with maximum response to the classifier $V(x)$ efficiently and fast, which is considered as vehicle tracking result in current UAV frame.

We also carefully check the vehicle tracking results, and the typical tracking outliers include large vehicle movements, stopped tracking vehicles, and neighboring vehicles' similar tracking positions. The former two typical tracking errors are likely to happen when vehicles are sheltered by obstacles in consecutive frames. The main reason is that the tracker fails to update vehicle tracking templates (i.e., obstacles are wrongly input as the vehicle cyclic samples). We have adjusted parameter settings in the KCF model to remove the tracking outliers. The outlier of neighboring vehicles' similar tracking positions usually happens when vehicle imaging size (the length or width of the vehicle image) is larger than predetermined vehicle detection area size (i.e., ROI length or width in each lane). We perform a data quality control procedure (to suppress the outlier) which merges two vehicles' positions into a large vehicle's position when the following constraints are met:

$$\begin{cases} \min\left\{\dfrac{S_o}{S_{v1}}, \dfrac{S_o}{S_{v2}}\right\} > \propto \\ O_f > N \end{cases} \tag{10}$$

where S_{v1} and S_{v2} are squares of the overlapped vehicles and S_o is square of the overlapped area. O_f is the total number of overlapped frames, and parameter $\propto$ and N are thresholds.

D. Coordinate Transformation of Vehicle Position

Vehicle positions in consecutive frames obtained in above steps are at the Cartesian coordinate positions (CCPs) which designate the x and y coordinate in the videos. However, traffic flow analysis requires vehicle trajectory data to be mapped at the Frenet coordinate positions (FCPs) [34]–[36]. The Frenet coordinates include the longitudinal (and lateral) coordinates along (and perpendicular to) roadways, where the former indicates vehicle's car following along the driving direction while the latter shows vehicle's lane change behavior. The coordinate transformation is briefly introduced here.

We sample points at each lane boundary in video frames, fit lane boundary curves, and then determine lane center curves (i.e., each lane is presented by the lane center curve). We can obtain the FCP longitudinal coordinates along roadways by mapping the CCPs to the lane center curves. As shown in Fig. 3, we employ $\{x(t), f(x(t))\}$, $(t = 0,1,2,..)$ to denote vehicle's CCPs (i.e. center point of red rectangles) at time t, and s_t $(t = 0,1,2,..)$ denotes the lane curve points which are closest to $\{x(t), f(x(t))\}$, $(t = 0,1,2,..)$. The FCP s_t (i.e., longitudinal coordinate) is the vehicle movement along trajectory. In fact, FCP s_t is the arc length between points s_t and s_{t+1}, which can be obtained by Eq. (11) and (12).

$$arc_{s_t s_{t+1}}(t) = \int_{x(t)}^{x(t+1)} f(x)dx \tag{11}$$

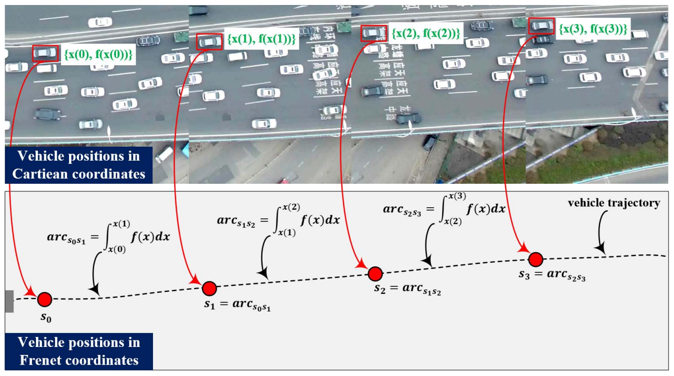

{x(0), f(x(0))}
{x(1), f(x(1))}
{x(2), f(x(2))}
{x(3), f(x(3))}
Vehicle positions in Cartiean coordinates
$arc_{s_0 s_1} = \int_{x(0)}^{x(1)} f(x)dx$
$arc_{s_1 s_2} = \int_{x(1)}^{x(2)} f(x)dx$
$arc_{s_2 s_3} = \int_{x(2)}^{x(3)} f(x)dx$
vehicle trajectory
s_0
$s_1 = arc_{s_0 s_1}$
$s_2 = arc_{s_1 s_2}$
$s_3 = arc_{s_2 s_3}$
Vehicle positions in Frenet coordinates

Fig. 3. Sketch map of transforming Cartesian coordinate to longitudinal Frenet coordinate.

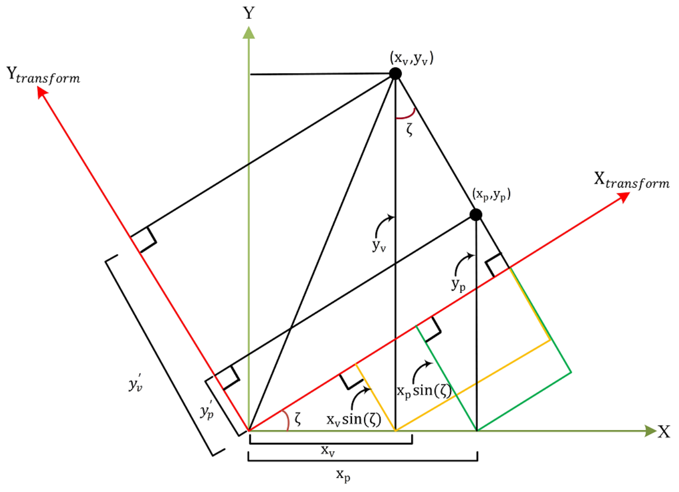


Fig. 4. Sketch map of transforming Cartesian coordinate to lateral Frenet coordinate.

$$s_{t+1} = s_t + arc_{s_t s_{t+1}}(t) \tag{12}$$

where $arc_{s_t s_{t+1}}$ is the arc length from point $s_s(t)$ to $s_{t+1}(t)$, $s_{s+1}(t)$ and $s_s(t)$ are the vehicle trajectory at time t.

The vehicle moving distance at the Frenet lateral coordinate can be calculated by the distance of vehicle positions and its projection on the lane center curve, with the coordinates transformed at an inclination angle of the projection point. Suppose point (x_v, y_v) and (x_p, y_p) are two vehicle positions and their projections on the lane center curve (see Fig. 4), respectively. The inclination angle of (x_p, y_p) is ζ. The transformed y_v is difference between $y_v\cos(\zeta)$ and $x_v\sin(\zeta)$ (see Eq. (13)). Similarly, we can obtain the transformed y_p by Eq. (14). The vehicle Frenet lateral coordinate L_t can be calculated by Eq. (15).

$$y_v' = y_v \cos(\zeta) - x_v \sin(\zeta) \tag{13}$$

$$y_p' = y_p \cos(\zeta) - x_v \sin(\zeta) \tag{14}$$

$$L_t = y_v' - y_p' \tag{15}$$

A main challenge in the vehicle position transformation is accurately fitting each lane curve in UAV frames. Considering UAV is hovering in the air when shooting the test videos (videos taken by moving UAV is beyond our discussion), we manually sample dozens of pixels in each lane boundaries, and employ the polyfit function [37] to fit upper and lower boundary curves for each lane, and obtain lane center curve by averaging coefficients of upper and lower boundaries. To find more accurate lane curves and suppress interference from randomly sampled pixels for lane curve fitting, we implement the procedure on each lane for several times (i.e., sample several groups of pixel points and obtain several groups of lane-curve fitting results), and manually find the optimal fitting curves as the final result for each lane.

E. Trajectory Denoising With Wavelet Transform

Though data quality control has been performed in each of the above procedures, only obvious errors are identified and corrected. There still exist small errors such as irregular oscillation of vehicle position caused by the vibration of tracking rectangle, due to the low video quality and background interference, which could affect the accuracy of vehicle trajectories. The WT is thus applied to eliminate such errors and the procedure is introduced in this section.

A WT filter decomposes raw trajectory data into scaling and wavelet subsets under a given basis. The wavelet subsets contain details and noises in the raw trajectory. The irregular oscillation in the raw trajectory results in high fluctuation margin of wavelet subsets, which can be considered as white noises. We can suppress these noises by setting up appropriate thresholds, and obtain clean trajectory by combining the scaling and non-noise wavelet trajectory subsets.

More specifically, the WT filter attempts to obtain clean trajectory data T_v^c from noisy trajectory data T_v^n in the following stages. First, a J-scale decomposition is applied to the T_v^n to obtain the approximate-trajectory part a_J and the detail-trajectory parts d_j (j = 1, …, J):

$$a_J = <T_v^n \cdot \varphi_J(t)>, \quad d_j = <T_v^n \cdot \psi_j(t)> \tag{16}$$

where $\varphi_J(t)$ and $\psi_j(t)$ are the wavelet basis function and scale function, respectively.

Since the trajectory approximate factor a_J is fixed, the trajectory detail factors d_j, (j = 1, …, J) will be smoothed by wavelet decomposition. Specifically, considering d_j, (j = 1, …, J) have different signal to noise ratios (SNRs), we set different thresholds for smoothing out noises in different trajectory detail signals. The rules of setting thresholds are given as follows.

As for the trajectory detail factor d_J, it contains most of the energy contributed to chaotic noises, and with higher SNR. Therefore, the threshold should not be too large to avoid removing the truly trajectory details. The threshold is set by Eq. (17):

$$t_J = \sigma_J \sqrt{2\ln N}/\sqrt{J} \tag{17}$$

As for the trajectory detail factor d_j, (j=1, 2 …J-1), there is a balance energy between chaotic signals and noise, and the threshold should be larger than t_J to successfully suppress noises in trajectory details, and the threshold is set as:

$$t_j = \sigma_J \sqrt{2\ln N}/\ln(j+1) \tag{18}$$

TABLE I
INFORMATION OF UAV VIDEOS

Information	Video #1	Video #2
Road Geometry	Curve	Curve
Frame rate	25 fps	23.98 fps
Traffic state	Free flow	Congested
Resolution	3840×2160	3840×2160
Duration	22 s	43 s
Focal length	23mm	23mm
Flying height	223m	281m

By smoothing noises in the trajectory detail factor d_j, (j = 1, 2 …J), we can obtain the smoothing trajectory T_v^c through Eq. (19):

$$T_v^c = a_J + \sum_{j=1}^{J} djj \tag{19}$$

where djj is the smoothed trajectory details of d_j.

III. Experiment Design

In data collection, the research team flied a UAV (model: DJI Mavic professional) above two urban expressway sections in Nanjing, China. The data collection covers both morning peak and off-peak periods on weekdays with good weather and visibility conditions. The detailed information of collected UAV videos are shown in table I. Both UAV videos are taken at a 3840 × 2160 resolution. Video #1 includes 550 frames taken at 25 frames-per-second. Video #2 includes 1032 frames taken at 23.98 frames-per-second. The UAV flying height for the two videos are 223m and 281m, respectively, and focal length are both 23 mm. Traffic states are free flow and congested, respectively. The methodological framework was developed on the MATLAB 2016 platform. The experimental tests were operated on a computer with an Intel i5-2310 CPU @ 2.9 GHz processor and a 6G memory.

The parameters in our methods were carefully determined to extract high-fidelity trajectory data. In the vehicle detection model, we tested various parameter settings and found the edge threshold C_{thresh}, histogram distribution threshold $H_{threshold}$ and structure element of Morphology operator $M_{threshold}$ had significant impacts on the ensemble detector's performance, and thus need to be set carefully. An example is given in Fig. 5 which shows the results with different parameter settings on vehicle detection. Comparing the leftmost snapshot with the right one in the same frame, a lower $C_{threshold}$ results in the detector obtaining a higher false-positive rate. Similarly, inappropriate settings of $H_{threshold}$ and $M_{threshold}$ mislead the vehicle detector. In our study, based on a fine trail-and-error criterion, the following parameter settings were selected: $C_{threshold} = 2\times10^{-1}$, $H_{threshold} = 50$ and $M_{threshold} = 5$.

Parameter setting in the vehicle-tracking module is more complicated. The reason is that the vehicle tracker employs the context of target vehicles (i.e. textures and contours from neighboring pixels) for determining the most possible region. We conducted the preliminary analyses and found that the

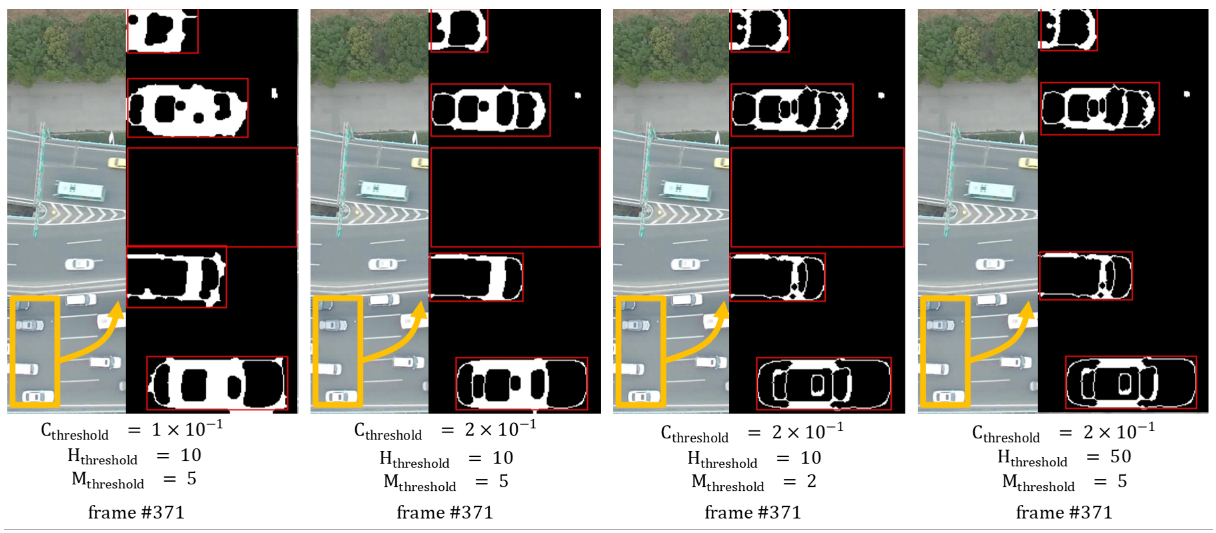


Fig. 5. Vehicle detection results with different parameter settings. (Black rectangle is vehicle detection ROI and red rectangle is detected vehicle positions).

three parameters, i.e. padding factor P_f, interpolation factor L_f and spatial bandwidth S_b, had significant impacts and needed to be carefully decided for obtaining a satisfied tracking performance. An example is given in Fig. 6 which shows the results with four sets of parameter settings. Comparing the leftmost snapshot with the right neighboring one in the same frame, a larger P_f results in the tracker tracking at a biased position. Similarly, inappropriate settings of L_f and S_b also mislead the vehicle tracker. In our study, the following parameter settings were finally used considering a tradeoff between accuracy and computing speed: $p_f = 1$, $L_f = 1.5\times10^{-2}$ and $S_b = 1\times10^{-1}$.

In the trajectory denoising model, we tested various parameter settings in the WT with commonly used basis families (which are Daubechies, Symlet, Coiflet). We found that the denoising performance was actually very robust. Considering the computation complexity of Daubechies basis is lower than those of the Symlet and Coiflet families, the Daubechies family was selected to smooth out outliers in the raw trajectory data. In our study, the db3 basis from Daubechies family was selected, and the level number was 3. The thresholds for level 1, 2, 3 were set to 2.510, 3.237, and 3.513, respectively.

IV. Experimental Results

A. *Performance Estimation Measures*

Vehicle trajectory data were extracted from the UAV videos using the proposed framework presented in the above sections. To validate the performance of our methods, the ground truth trajectories were manually extracted by four graduate students from the raw UAV videos. More specifically, we manually marked the front and rear bumper position of each vehicle in initial frames by drawing the detector box, and continuously monitored and corrected each vehicle's position in each time step of the tracking procedure in order to make sure each vehicle's ground truth trajectory is accurate.

Three measures of fitting goodness are considered for evaluating the accuracy of vehicle trajectory extraction, which are the Root-mean-square deviation (RMSE), the Mean Squared Deviation (MSD), and the Pearson product-moment correlation coefficient (Pearson's r). The indicators are commonly used for comparing the fitting data with the reference data [38]. For each vehicle trajectory, statistical indices are computed from Eq. (20) to (24). A smaller value of RMSE, MSD or larger Pearson's r indicates that the trajectory data is closer to the true value, and vice versa.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum\nolimits_{t=1}^{n} \left|T_{smth}(t) - T_{Grud}(t)\right|} \tag{20}$$

$$\mathrm{MSD} = \frac{\sum\nolimits_{t=1}^{n} \left|T_{smth}(t) - T_{Grud}(t)\right|^2}{n} \tag{21}$$

$$\mathrm{Pearson's\ r} = \frac{\sum\nolimits_{t=1}^{n}\left[\left(T_{smth}(t) - \overline{T_{smth}}\right)\times\left(T_{Grud}(t) - \overline{T_{Grud}}\right)\right]}{\sqrt{\sum\nolimits_{t=1}^{n}\left(T_{smth}(t) - \overline{T_{smth}}\right)^2}\times\sqrt{\sum\nolimits_{t=1}^{n}\left(T_{Grud}(t) - \overline{T_{Grud}}\right)^2}} \tag{22}$$

$$\overline{T_{smth}} = \frac{1}{n}\sum\nolimits_{t=1}^{n} T_{smth}(t) \tag{23}$$

$$\overline{T_{Grud}} = \frac{1}{n}\sum\nolimits_{t=1}^{n} T_{Grud}(t) \tag{24}$$

where t is trajectory point number, $T_{smth}(t)$ and $T_{Grud}(t)$ are the smoothed and ground truth trajectory data points, $\bar{T}_{smth}$ and $\bar{T}_{Grud}$ are the mean values of $T_{smth}(t)$ and $T_{Grud}(t)$, and n is the number of sample.

B. *Outputs of Methodological Procedures*

The outputs from each step of the methodological framework on video #1 were presented in detail to show how our models work. Typical vehicle detection examples are shown in Fig. 7. It is seen that true vehicles in the ROI are detected successfully, though detection outliers (i.e., empty box, small box, etc.) exist before conducting the data quality control. The

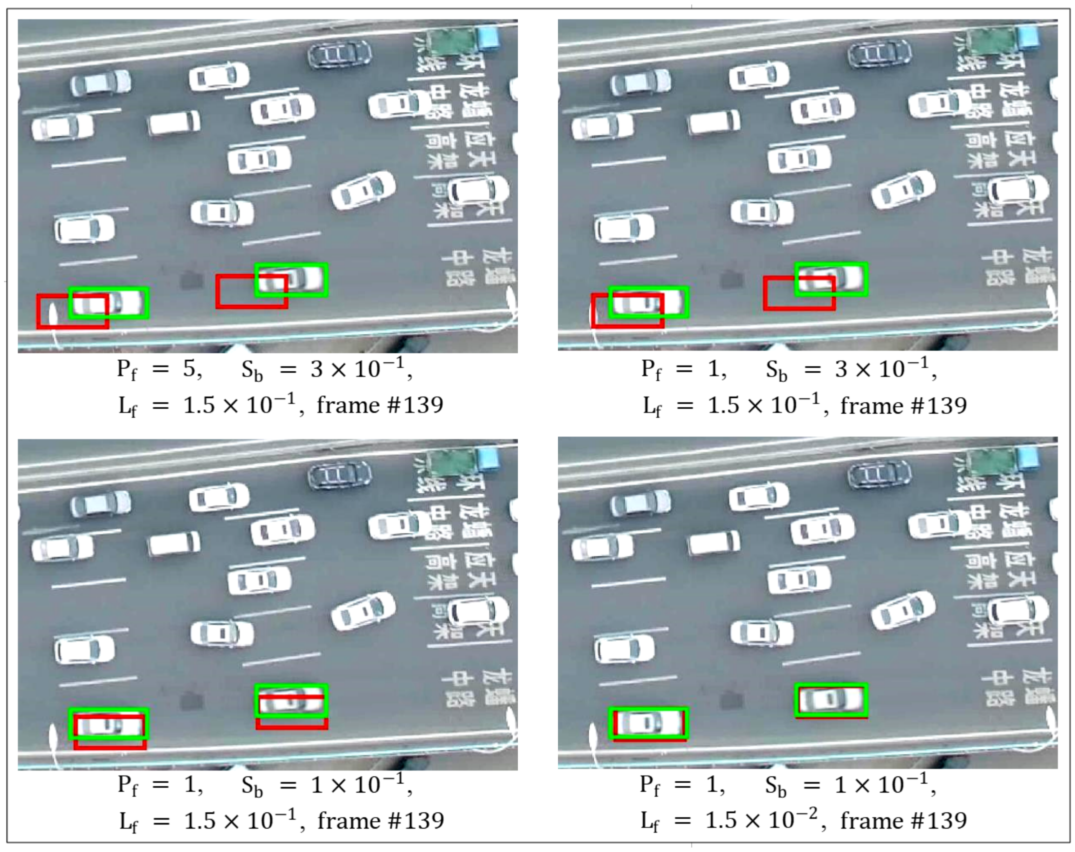

Pf = 5, Sb = 3 × 10^−1, Lf = 1.5 × 10^−1, frame #139
Pf = 1, Sb = 3 × 10^−1, Lf = 1.5 × 10^−1, frame #139
Pf = 1, Sb = 1 × 10^−1, Lf = 1.5 × 10^−1, frame #139
Pf = 1, Sb = 1 × 10^−1, Lf = 1.5 × 10^−2, frame #139

Fig. 6. Vehicle tracking results with different parameter settings. (Green rectangle is vehicle tracking position and red rectangle is ground truth position).

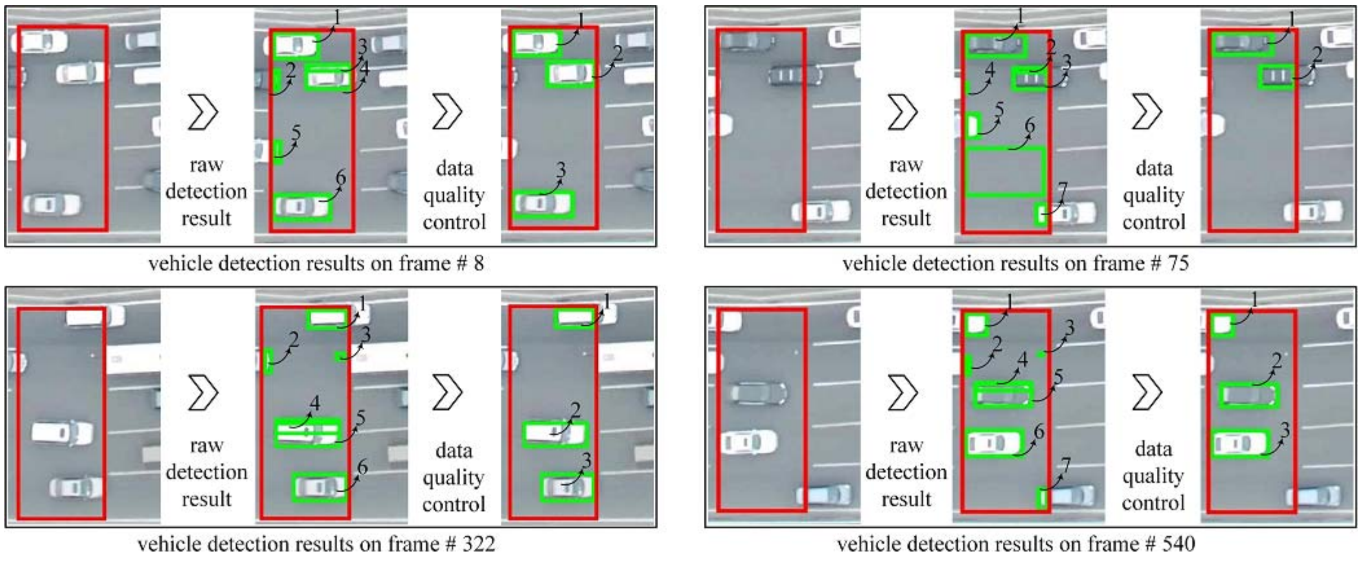

raw detection result
data quality control
vehicle detection results on frame # 8
vehicle detection results on frame # 75
vehicle detection results on frame # 322
vehicle detection results on frame # 540

Fig. 7. Results of vehicle detection by the ensemble detector.

main reason for the empty box (see frame # 75 in Fig. 7) is found to be that border pixel intensity of each lane is slightly larger than other pixel intensity as they are close to lane separator. If there is no vehicle in the detection zone, the lane border may be erroneously identified as vehicle edge. In the data quality control process, such empty-box outliers are successfully discarded by designating the threshold of intensity distribution range (see Fig. 7). The small box issue does not affect the result because when a vehicle body comes into the ROI, the algorithm will update the detector box and use the maximum one for the tracking purpose. After the data quality control, our ensemble model recognizes all vehicles in the ROI in each frame.

An example of vehicle tracking result is shown in Fig. 8. The tracking vehicle number in the plots starts from 1 for simplicity. It is noticed that in the raw tracking results, as shown in left snapshots, a bus is tracked as two small cars (see the 4th and 5th vehicle in the left snapshot of Fig. 8(a))

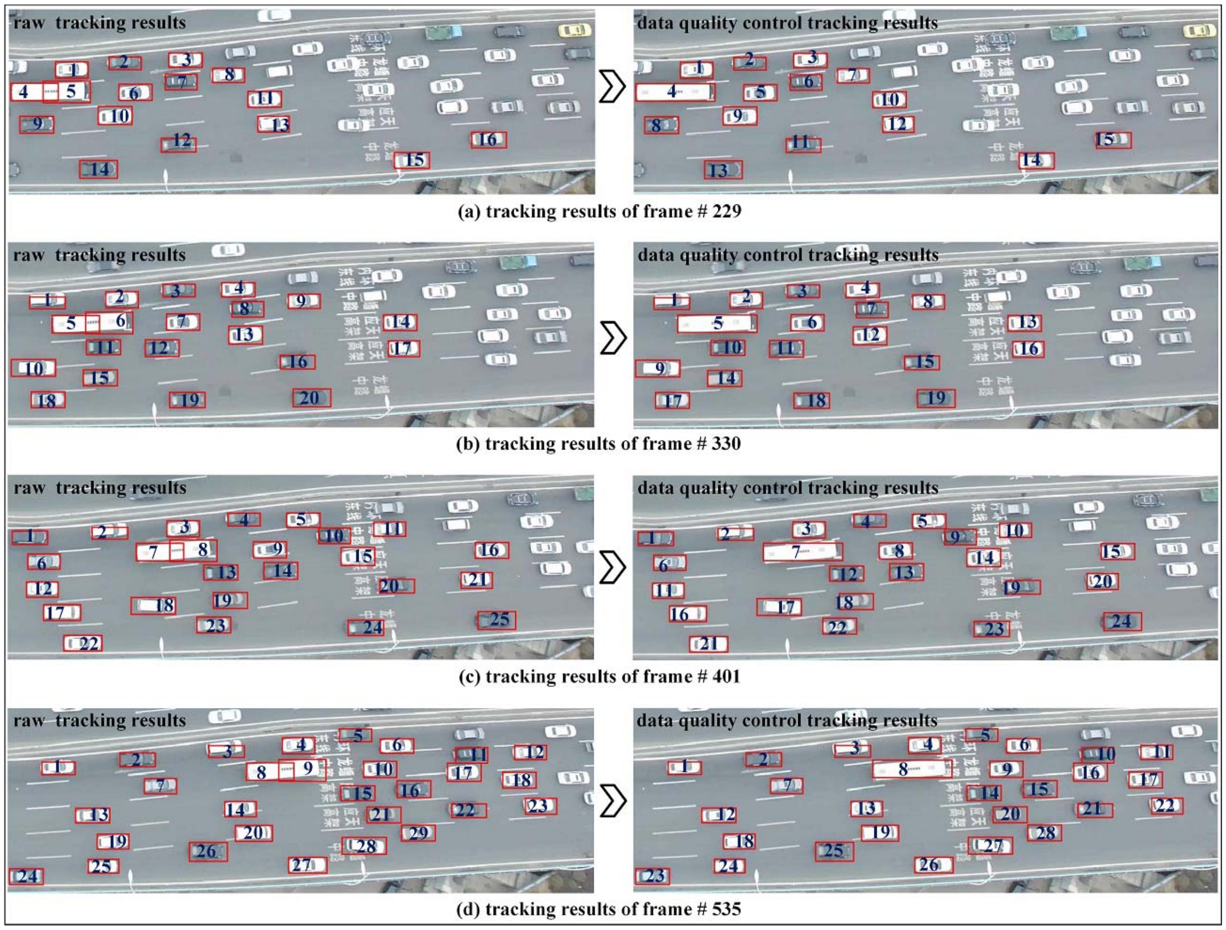

raw tracking results
data quality control tracking results
(a) tracking results of frame # 229
(b) tracking results of frame # 330
(c) tracking results of frame # 401
(d) tracking results of frame # 535

Fig. 8. Results of vehicle tracking by the KCF model.

as its length exceeds the ROI length in the lane. The data quality control identifies the outlier based on the rule that the two vehicles' positions intersect with each other in numerous frames, and thus suppresses the outlier by merging them into one large vehicle (see the right snapshot of Fig. 8(a)). Besides, the 1st vehicle in the frame #330 (i.e., the 2nd vehicle in the frame #401, and the 3rd vehicle in the frame #535) is not tracked 100% accurately. The reason is that the predetermined detecting ROI is manually labelled which excludes each lane boundary pixels in ROI (for the purpose of suppressing roadway boundary interference in detection procedure), and thus the vehicle pixels overlapping with roadway boundaries cannot be detected by our framework, resulting in the KCF tracker being initialized with incomplete vehicle pixels.

Based on vehicle positions and fitted lane curve information, we mapped vehicle position in video into vehicle trajectory on road. The WT was then employed to remove trivial noises and outliers in the raw trajectory data. Examples of trajectories are shown in Fig. 9. The trajectory of the first detected vehicle in lane #1 is labelled as car #1 raw, smoothed, and ground truth trajectory. The principle is applicable to car #2, #3, #4, #5. Though the raw vehicle trajectory data are quite close to the ground-truth trajectories, inconsistences between raw and the ground-truth data can be observed when zooming into details (see Fig. 9). It can be seen that the smoothed trajectory data are closer to the ground truth data, which demonstrates the importance and effectiveness of the WT denoising procedure.

C. Results of Vehicle Trajectory Extraction

We performed the procedures on all frames in video #1 and estimated vehicle trajectories with the time interval of 0.04s, 0.2s and 0.4s, respectively (i.e. trajectory was obtained per frame, 5 frame and 10 frame). We compared the raw and denoised trajectory with the ground truth data, and the results are summarized in Table II. It is found that the processed trajectory data are obviously more accurate (i.e. closer to the true data) than the raw data without denoising procedure, in terms of smaller MSD and RMSE values. There is no large difference in the Pearson's r as the values are quite close to 1. We also notice that the time interval for trajectory estimation does not significantly affect the data accuracy (see Table II). It indicates that our extracted trajectory data are very accurate even at a high-resolution of time interval.

The proposed framework was applied on the video #2 where traffic was congested, and time interval of 0.04 s was applied for estimating the trajectory extraction accuracy. The results are shown in Table III. It is found that trajectory accuracy in video # 2 is lower than that in video # 1. A possible reason is that UAV flies at a higher altitude in video # 2. Consequently,

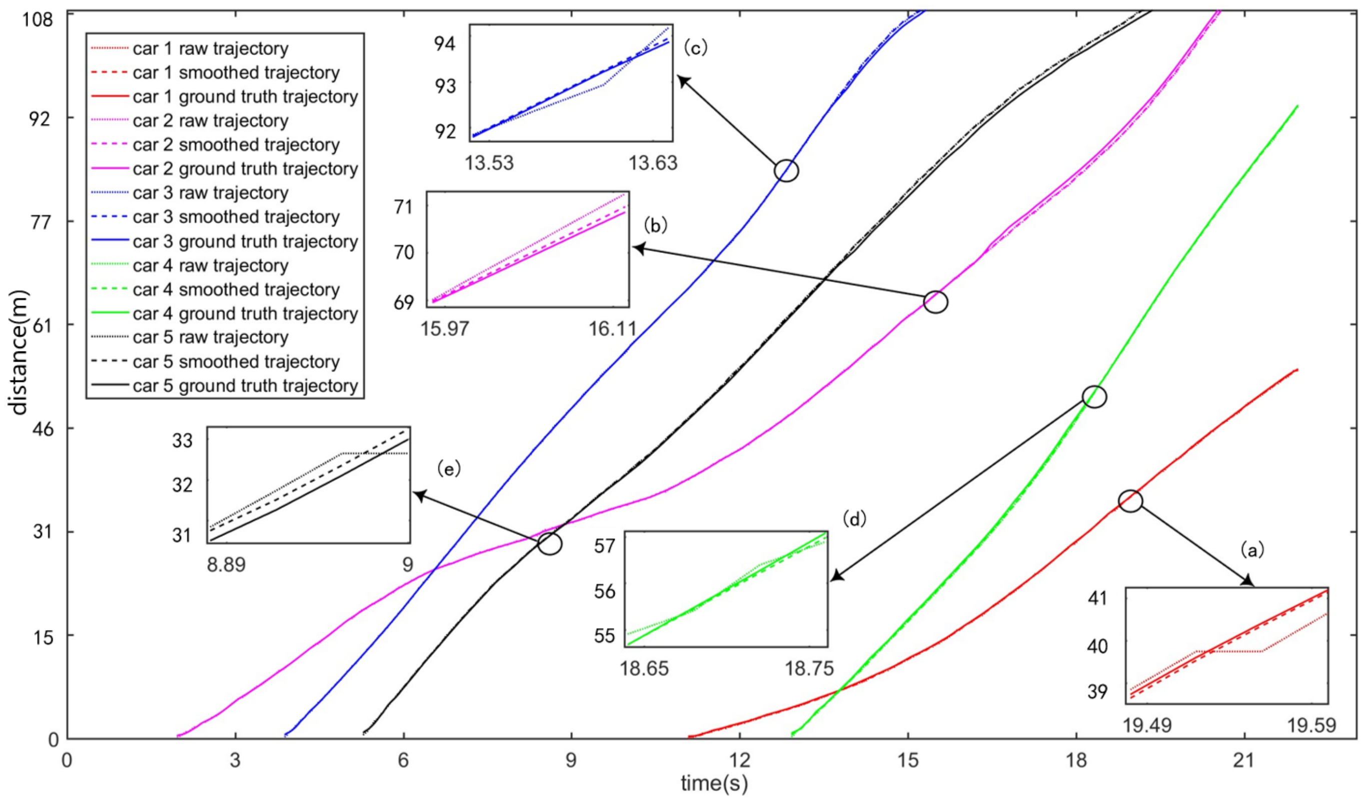


Fig. 9. Results of vehicle trajectory denoising with the WT method.

TABLE II
STATISTICAL PERFORMANCE OF VEHICLE TRAJECTORIES AT DIFFERENT FRAME INTERVALS FOR VIDEO # 1

Traffic parameter	MSD (m)		RMSE (m)		Pearson' r	
	Raw vs. Ground Truth	Smooth vs. Ground Truth	Raw vs. Ground Truth	Smooth vs. Ground Truth	Raw vs. Ground Truth	Smooth vs. Ground Truth
Time interval of 0.04 s						
Trajectory of lane 1	0.588	0.501	0.150	0.100	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 2	0.754	0.654	0.208	0.167	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 3	0.347	0.228	0.139	0.597	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 4	1.073	0.933	0.246	0.185	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 5	0.106	0.935	0.255	0.207	9.999×10^{-1}	9.999×10^{-1}
Average	0.765	**0.650**	0.199	**0.144**	9.999×10^{-1}	9.999×10^{-1}
Time interval of 0.2 s						
Trajectory of lane 1	0.224	0.019	0.125	0.024	9.998×10^{-1}	9.999×10^{-1}
Trajectory of lane 2	0.911	0.710	0.250	0.189	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 3	0.582	0.332	0.185	0.081	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 4	0.873	0.620	0.224	0.152	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 5	1.835	0.158	0.352	0.282	9.999×10^{-1}	9.999×10^{-1}
Average	0.885	**0.652**	0.227	**0.146**	9.999×10^{-1}	9.999×10^{-1}
Time interval of 0.4 s						
Trajectory of lane 1	0.363	0.033	0.157	0.032	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 2	1.162	0.844	0.286	0.207	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 3	0.683	0.350	0.202	0.081	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 4	1.038	0.695	0.248	0.162	9.999×10^{-1}	9.999×10^{-1}
Trajectory of lane 5	0.190	1.623	0.359	0.287	9.999×10^{-1}	9.999×10^{-1}
Average	1.029	**0.709**	0.251	**0.154**	9.999×10^{-1}	9.999×10^{-1}

the trajectory extraction accuracy in video # 2 is more sensitive to minor measurement errors of vehicle positions in UAV videos.

The extracted vehicle trajectories in video #2 are shown in Fig. 10, supporting a variety of traffic flow analysis. We find that lane #1, #2 and #4 have more discontinued vehicle trajectories than the other two lanes, which means drivers in the three lanes are more likely to change lanes during the surveillance time spans. The lane change behavior happens in lanes with larger traffic speed (i.e., lane #1, #2), indicating that drivers prefer to change lanes and overtake neighbors for the sake of moving out congested areas fast. We can also estimate

TABLE III
STATISTICAL PERFORMANCE OF VEHICLE TRAJECTORIES AT 0.04S INTERVAL FOR VIDEO # 2
(NOTE: THERE IS NO LARGE DIFFERENCE IN THE PEARSON'S R VALUES)

Traffic parameter	MSD (m)		RMSE (m)	
	Raw vs. Ground Truth	Smooth vs. Ground Truth	Raw vs. Ground Truth	Smooth vs. Ground Truth
Trajectory of lane 1	3.575	3.461	0.267	0.197
Trajectory of lane 2	2.549	2.422	0.272	0.188
Trajectory of lane 3	1.898	1.77	0.229	0.142
Trajectory of lane 4	8.377	8.236	0.390	0.305
Trajectory of lane 5	4.085	3.943	0.2930	0.199
Average	4.097	3.967	0.2903	0.206

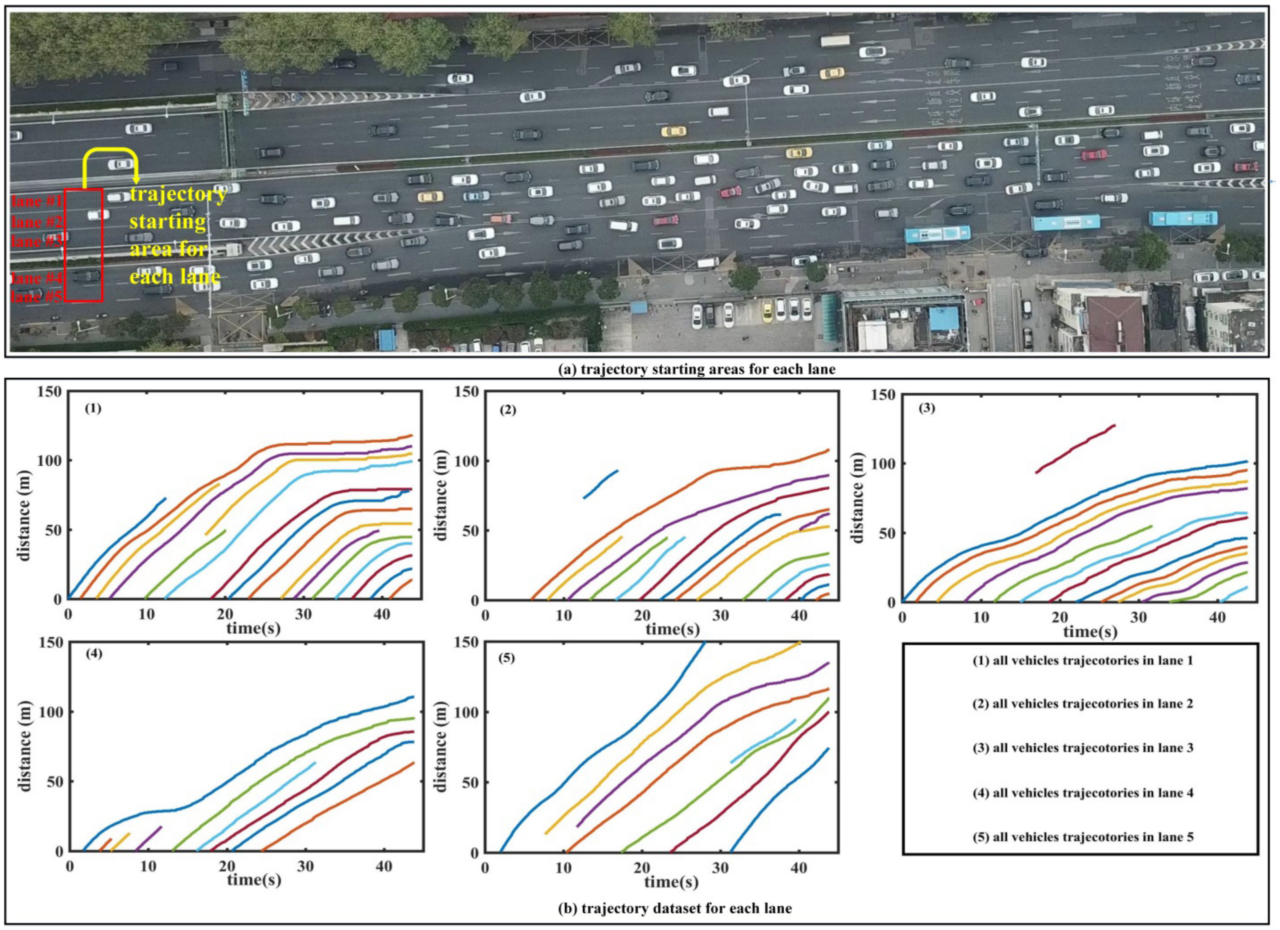


Fig. 10. Extracted vehicle trajectories on different lanes in video # 2.

average time headway for each lane based on the extracted trajectory data. The approximate average time headway for lane #1, #2, #3, #4 and 5 are 2.404s, 2.148s, 2.109s, 3.371s and 3.457s, respectively. The average time headway is 2.698s, indicating the average traffic flow is about 1335 pcu/h/lane.

V. CONCLUSION AND FUTURE WORK

In this study, we proposed a methodological framework based on the computer vision techniques for extracting high-resolution vehicle trajectories automatically from videos taken by bird-view UAV cameras. The first procedure was the vehicle detection in ROI in consecutive frames with a Canny based ensemble edge detector. The second procedure was the robust and fast vehicle tracking by a KCF tracker which has gained high tracking accuracy and speed. The third procedure transformed coordinates by mapping vehicle positions in UAV videos from the Cartesian coordinate to Frenet coordinate, and extracted vehicle trajectories by accumulating the transformed longitudinal Frenet coordinates. Data quality control has been integrated in each step to suppress obvious errors, and obtain accurate vehicle positions. The final step was to eliminate noises in the raw trajectories with the WT filter.

The extracted vehicle trajectories were compared with those from manual identifications. The results showed that the extracted trajectories were reasonably close to the ground truth data. We also presented the time-space trajectory map based on the extracted data and discussed traffic flow characteristics. The methodology proposed in this study can support the efficient and accurate extraction of vehicle trajectories from UAV videos, which could greatly enrich trajectory dataset in more traffic conditions for traffic flow studies. We made the video and trajectory data publicly accessible for benchmark at *https://seutraffic.com.*

In our future research, we plan to expand our current work by considering the following directions. First, in the current UAV videos, the camera is set at a bird-view angle and does not move during the data collection period. We could enhance the usability of the proposed vehicle trajectory extraction algorithms under challenging situations such as moving UAV with multi-dimensional camera motions (rolling, heaving and surging, combination of the two motions, etc.). Second, traffic state in the current UAV videos is relatively stable and homogeneous. We could investigate the performance of our approaches for other traffic states or complex state transition status to improve the practicability of our methods. Third, current UAV videos are taken in clear visibility conditions. We could enhance the framework and improve the model performance for poor visibility situations such as night time, raining, snowing, etc. Last but not least, we could employ deep learning models for accurate vehicle detections and assemble vehicle trajectories by solving the vehicle matching tasks in different image frames. Authors recommend future studies may focus on the above issues.

References

[1] V. Punzo, B. Ciuffo, and M. Montanino, "Can results of car-following model calibration based on trajectory data be trusted?" *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2315, no. 1, pp. 11–24, Jan. 2012.

[2] J. Monteil, R. Billot, J. Sau, C. Buisson, and N.-E.-E. Faouzi, "Calibration, estimation, and sampling issues of car-following parameters," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2422, no. 1, pp. 131–140, Jan. 2014.

[3] A. Sopasakis and M. A. Katsoulakis, "Information metrics for improved traffic model fidelity through sensitivity analysis and data assimilation," *Transp. Res. B, Methodol.*, vol. 86, pp. 1–18, Apr. 2016.

[4] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 12, pp. 2159–2168, Dec. 2013.

[5] B. Coifman, "Empirical flow-density and speed-spacing relationships: Evidence of vehicle length dependency," *Transp. Res. B, Methodol.*, vol. 78, pp. 54–65, Aug. 2015.

[6] C. P. Schwegmann, W. Kleynhans, and B. P. Salmon, "Synthetic aperture radar ship detection using Haar-like features," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 2, pp. 154–158, Feb. 2017.

[7] Q. Li, W. Zhang, M. Li, J. Niu, and Q. M. Jonathan Wu, "Automatic detection of ship targets based on wavelet transform for HF surface wavelet radar," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 714–718, May 2017.

[8] X. Zhao, Q. Li, D. Xie, J. Bi, R. Lu, and C. Li, "Risk perception and the warning strategy based on microscopic driving state," *Accident Anal. Prevention*, vol. 118, pp. 154–165, Sep. 2018.

[9] X. Wang, J. Zhang, Y. Liu, W. Yunyun, F. Wang, and J. Wang, "The drivers' lane selection model based on mixed fuzzy many-person multi-objective non-cooperative game," *J. Intell. Fuzzy Syst.*, vol. 32, no. 6, pp. 4235–4246, May 2017.

[10] S. Tak, S. Kim, and H. Yeo, "A study on the traffic predictive cruise control strategy with downstream traffic information," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1932–1943, Jul. 2016.

[11] E. Balal, R. L. Cheu, and T. Sarkodie-Gyan, "A binary decision model for discretionary lane changing move based on fuzzy inference system," *Transp. Res. C, Emerg. Technol.*, vol. 67, pp. 47–61, Jun. 2016.

[12] V. G. Kovvali and V. P. Alexiadis Zhang, "Video-based vehicle trajectory data collection," in *Proc. 86th Annu. Meeting Transp. Res. Board*, Washington, DC, USA, 2007, p. 0528.

[13] Z. He, *Research Based on High-Fidelity NGSIM Vehicle Trajectory Datasets: A Review*. Berlin, Germany: Reseachgate, 2017, pp. 1–33.

[14] B. Coifman and L. Li, "A critical evaluation of the next generation simulation (NGSIM) vehicle trajectory dataset," *Transp. Res. B, Methodol.*, vol. 105, pp. 362–377, Apr. 2017.

[15] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transp. Res. B, Methodol.*, vol. 80, pp. 82–106, Oct. 2015.

[16] E. N. Barmpounakis, E. I. Vlahogianni, and J. C. Golias, "Unmanned aerial aircraft systems for transportation engineering: Current practice and future challenges," *Int. J. Transp. Sci. Technol.*, vol. 5, no. 3, pp. 111–122, Oct. 2016.

[17] C. L. Azevedo, J. L. Cardoso, M. Ben-Akiva, J. P. Costeira, and M. Marques, "Automatic vehicle trajectory extraction by aerial remote sensing," *Proc.—Social Behav. Sci.*, vol. 111, pp. 849–858, Feb. 2014.

[18] R. Ke, Z. Li, J. Tang, Z. Pan, and Y. Wang, "Real-time traffic flow parameter estimation from UAV video based on ensemble classifier and optical flow," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 54–64, Jan. 2019.

[19] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-time bidirectional traffic flow parameter estimation from aerial videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 890–901, Apr. 2017.

[20] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transp. Res. C, Emerg. Technol.*, vol. 6, no. 4, pp. 271–288, Aug. 1998.

[21] G. Guido, V. Gallelli, D. Rogano, and A. Vitale, "Evaluating the accuracy of vehicle tracking data obtained from unmanned aerial vehicles," *Int. J. Transp. Sci. Technol.*, vol. 5, no. 3, pp. 136–151, Oct. 2016.

[22] A. C. Shastry and R. A. Schowengerdt, "Airborne video registration and traffic-flow parameter estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 4, pp. 391–405, Dec. 2005.

[23] T. Moranduzzo and F. Melgani, "Automatic car counting method for unmanned aerial vehicle images," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1635–1647, Mar. 2014.

[24] R. Girdhar, "Simple, efficient and effective keypoint tracking," in *Proc. ICCV*, 2017. [Online]. Available: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Simple%2C+efficient+and+effective+keypoint+tracking&btnG=

[25] Y. Li, J. Zhu, and S. C. H. Hoi, "Reliable patch trackers: Robust visual tracking by exploiting reliable patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 353–361.

[26] M. Cen and C. Jung, "Fully convolutional siamese fusion networks for object tracking," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3718–3722.

[27] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[28] X. Chen, S. Wang, C. Shi, H. Wu, J. Zhao, and J. Fu, "Robust ship tracking via multi-view learning and sparse representation," *J. Navigat.*, vol. 72, no. 1, pp. 176–192, Jan. 2019.

[29] T. Zhou, X. He, K. Xie, K. Fu, J. Zhang, and J. Yang, "Robust visual tracking via efficient manifold ranking with low-dimensional compressive features," *Pattern Recognit.*, vol. 48, no. 8, pp. 2459–2473, Aug. 2015.

[30] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[31] Y. Zhang, T. Y. Ji, M. S. Li, and Q. H. Wu, "Identification of power disturbances using generalized morphological open-closing and close-opening undecimated wavelet," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2330–2339, Apr. 2016.

[32] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[33] R. M. Gray, "Toeplitz and circulant matrices: A review," *Found. Trends Commun. Inf. Theory*, vol. 2, no. 3, pp. 155–239, 2005.

[34] Z. Whan Kim and J. Malik, "High-quality vehicle trajectory generation from video data based on vehicle detection and description," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Oct. 2003, pp. 176–182.

[35] M. Trivedi, "Understanding vehicular traffic behavior from video: A survey of unsupervised approaches," *J. Electron. Imag.*, vol. 22, no. 4, p. 1113, 2013.

[36] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, Mar. 2012.

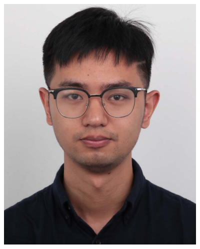
[37] J.-W. Lu, Y.-J. He, H.-Y. Li, and F.-L. Lu, "Detecting small target of ship at sea by infrared image," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Oct. 2006, pp. 165–169.

[38] B. Xie, L. Hu, and W. Mu, "Background suppression based on improved top-hat and saliency map filtering for infrared ship detection," in *Proc. Int. Conf. Comput. Intell. Inf. Syst. (CIIS)*, Apr. 2017, pp. 298–301.

Xinqiang Chen (Member, IEEE) received the Ph.D. degree in traffic information engineering and control from Shanghai Maritime University, China, in 2018. From September 2015 to September 2016, he was a Visiting Student with the Smart Transportation Applications and Research Laboratory, University of Washington, USA. He is the author and coauthor of more than 13 technical articles. His research interests include traffic data analysis, transportation image processing, transportation video analysis, and smart ship.

Zhibin Li received the Ph.D. degree from the School of Transportation, Southeast University, China, in 2014. From 2015 to 2017, he was a Post-Doctoral Researcher with the University of Washington and The Hong Kong Polytechnic University. From 2010 to 2012, he was a Visiting Student with the University of California, Berkeley. He is currently a Professor with Southeast University. His research interests include intelligent transportation, traffic safety, data mining, traffic control, artificial intelligence, and so on.

Yongsheng Yang (Member, IEEE) received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics, China, in 1998. He is currently a Professor with Shanghai Maritime University. His research interests include port logistics operation and optimization, and cooperated job scheduling and controlling for automatic terminals. He also serves as an Associate Editor for the *Journal of Computer Aided Engineering*.



Lei Qi received the master's degree from the School of Computer Science and Engineering, Nanjing University of Science and Technology, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University, China. He was a Visiting Student with the University of Wollongong, from August 2018 to August 2019. His research interests include data mining, computer vision, machine learning, multimedia, and visual surveillance. His work mainly focuses on the person re-identification task in video surveillance systems.

Ruimin Ke (Member, IEEE) received the B.E. degree from the Department of Automation, Tsinghua University, in 2014, and the M.S. degree from the Civil and Environmental Engineering, University of Washington, in 2016, where he is currently pursuing the Ph.D. degree in civil and environmental engineering His research interests include intelligent transportation systems, transportation data science, smart city, autonomous driving, the Internet of Things, and computer vision. He is a member of the Statewide and National Data and Information Management Committee of the TRB, a Young Member of the Infrastructure Systems Committee of ASCE T&DI, and a member of the Urban Computing and Space Optimization Committee of WTC.