Neural Computing and Applications (2023) 35:8747–8761
https://doi.org/10.1007/s00521-022-07570-1

S.I.: AI BASED TECHNIQUES AND APPLICATIONS FOR INTELLIGENT IOT SYSTEMS

Bidirectional feedback of optimized gaussian mixture model and kernel correlation filter for enhancing simple detection of small pixel vehicles

Xiaofeng Shan[1] · Qifan Wu[2] · Zhibin Li[2] · Chishe Wang[1]

Received: 7 March 2022 / Accepted: 27 June 2022 / Published online: 31 July 2022

Abstract
High-precision detection of vehicle position and contour from unmanned aerial vehicles (UAV) provides critical information for vehicle behavior and traffic flow studies. Vehicles in UAV videos present unique features of small target pixels, which pose challenges in accurate detection. In addition, shaking of UAV camera, shadow of vehicle, and ground sign/marking also lead to difficulties in precise vehicle contour detection. The study proposes a novel approach that designs a bidirectional feedback framework (GKB) between optimized Gaussian mixture model and Kernel correlation filter to enhance vehicle detection. The framework predicts vehicle position based on information of continuous and correlated previous frames to achieve improved performance. We also improve the detection of closely spaced and dark vehicles with morphological algorithms and data processing. The approach is tested on two UAV videos with different shooting heights, illumination conditions, and traffic states. The results show that the proposed method significantly improves vehicle detection. The total accuracy of our model is 98%, which is a 11% improvement over the traditional single detect model and a 4% improvement over the track-after-detect method. Our model's detection rate of closely spaced and dark vehicles is improved by 15–25% compared to previous methods. Our model's vehicle contour detection accuracy is over 94%, which is about a 15% improvement over previous methods.

Keywords Unmanned aerial vehicle · Vehicle detection · Bidirectional feedback · Accuracy position

1 Introduction

High-precision vehicle detection from videos benefits traffic flow studies by generating vehicle trajectories containing information such as vehicle position, speed, acceleration, and gap. They greatly support traffic flow modeling, traffic congestion analysis, and traffic conflict evaluation [1–3]. In recent years, due to the advantages of comprehensive visibility coverage, low deployment cost, and high flexibility, the unmanned aerial vehicle (UAV) is becoming an emerging technology for collecting traffic videos. However, vehicles in UAV videos present some unique features such as small targets and low pixels, which pose challenges in foreground extraction and vehicle-background distinguishing and thus usually result in the decreased detection rate of vehicles. In addition, shaking of UAV camera, shadow of vehicle, and ground sign/marking also lead to difficulties in accurately detecting vehicle contours such as front and rear bumpers. Improving vehicle detection rate in UAV videos is a challenging research topic [4–6].

Currently, the hottest method of vehicle detection is deep learning. Those models mainly include supervised learning (such as convolutional neural network (CNN), fast region convolutional neural network (FAST-RCNN), Yolo

✉ Qifan Wu
wuqifan@seu.edu.cn

✉ Zhibin Li
lizhibin@seu.edu.cn

[1] Jinling Institute of Technology, 99 Hongjing Road, Nanjing 211169, China

[2] Southeast University, 2 Si Pai Lou, Nanjing 211169, China

[7–9]) and semi-supervised or unsupervised learning (such as generative adversarial nets (GANs), stacked capsule autoencoder (SCAE) [10, 11]). A reasonable vehicle detection rate can be achieved based on the condition that the deep learning agent has been well-trained by a relatively large number of sample sets. In other words, the performance is overly dependent on the training sample quality and can be disturbed by factors prone to underfitting and over-fitting phenomena in different scenes. The training also requires a lot of work such as preparation, parameters tuning, time, and environmental allocation. In scenarios without sufficient and high-quality training samples, such as our case of vehicle detection with small-target, low-resolution, changeable condition in UAV videos, the deep learning model performance can be primarily limited [12–14].

Previous studies have proposed machine vision methods for vehicle detection, such as the adaptive Gaussian mixture model [15], background modeling for foreground detection [16], Harris corner detection method [17], and universal background subtraction vibe algorithm [18]. The central idea of those methods is establishing the video background and finding the features of moving objects in pixel variety for target detection. Compared with the deep learning methods, the advantages are no need for an extensive training sample set and heavy preparation work, which is robust to new scenarios. However, the background models are highly influenced by video shaking and scene interference, and the foreground extraction is greatly affected by the vehicle pixels. As a result, the model performance usually arises the minuses such as vehicle shadow misdetection, abnormal connection of closely spaced vehicles, and ghost areas by invalid foreground [19].

In recent years, target tracking algorithms have been applied to track vehicles' positions in UAV videos. For example, Ke et al. develop a Kanade–Lucas–Tomasi tracking method [20]. Kristan et al. developed a visual object tracking method [21]. Lee et al. developed a visual tracking method by partition-based histogram back-projection and maximum support criteria [22]. Chen et al. develop a high-resolution vehicle trajectory extraction method using the region of intersect (ROI) detection and the Kernel correlation filter (KCF) tracking [23]. Ren et al. propose state-of-the-art multi-object tracking (MOT) methods to obtain the trajectories [24]. Naima Amrouche et al. developed a track-before-detect (TBD) approach [25]. The main idea of these algorithms is to predict the subsequent positions of vehicles through pixel features from previous positions. They improve the robustness and accuracy of detecting the to-be-tracked vehicles. However, most tracking algorithms predict the target position only from the function of extreme value acquisition by ridge regression and other methods, which may cause the tracking model to be easily influenced by changeable conditions and neighboring vehicles. As a result, these tracking algorithms often suffer from the problem of miss detection and loss tracking during the detection of complex scenes [26].

The primary objective of our study is to propose a bidirectional feedback framework for the optimized Gaussian mixture model (OGMM) and Kernel correlation filter (KCF) by optimizing the interaction to enhance the performance of vehicle detection. The framework uses a generative complementarity (GC) structure through the correlated-frame motion characteristics to optimize the modeling and improve the detection of vehicle and contour. The results are compared with other models for different scenarios. The findings of our study can benefit vehicle trajectory extraction, traffic flow modeling, traffic incident detection, and vehicle sample database building-up.

2 Methodology

2.1 Overall framework

We proposed the bidirectional feedback framework (GKB) between optimized Gaussian mixture model (OGMM) and Kernel correlation filter (KCF) to obtain high-precision trajectory data and vehicle contour data from UAV video, especially for the detection of small pixel targets and closely spaced vehicle. The GKB framework is derived from the generative complementarity thought, in which the two methods (detection and track) can play to their strength while complementing each other's disadvantages by the feedback mechanism. We designed our detection model with solid robustness to better detect UAV video in complex and changeable scenes without enough sample training. The core steps of the algorithm include: (i). Pretreatment: We used our new pixel feature enhancement framework, which provides for scale-invariant feature transform (SIFT), feature extraction (FE), linear affine (LA), and k-nearest neighbor (KNN), to eliminate background interference (including jitter, light, and shadow) and optimize the background image and foreground pixel. (ii). Detection and Tracking: We adapted the GKB framework, which enhances detection through inter-frame position features. Meanwhile, the framework proposes coordinate information optimization based on two-way feedback, solving some detection problems of traditional algorithms, such as missed inspection and the 'ghost Region of Intersect (ROI)' area. (iii). Data Processing and Optimization: We optimized the new trajectory by coordinating accurate regression and abnormal trajectory

elimination to collect high-precision trajectory data. Details of the algorithm are given as follows.

3 Notations

Before formulating the GKB model, the notations used in this paper are listed in Table 1.

3.1 Pretreatment

To extract the foreground target effectively when the UAV video background is instability, we propose a new framework based on scale-invariant feature transform (SIFT) feature extraction, *k*-nearest neighbor (KNN) matching, and linear affine transform [27, 28]. Firstly, we will use SIFT algorithm to extract the feature points in aerial video and obtain their scale vector information. Secondly, we use the KNN proximity method to match the feature points between interference frames and map the jitter frame back to the current standard frame through the feature linear affine transformation matrix to eliminate the jitter of UAV. Finally, we will enhance the target area of the foreground through binarization algorithm and opening and closing operation, which will help us better carry out video detection.

3.1.1 Background optimization method

In the beginning, we use the SIFT method to create a new Gauss-scale-space to represent the video background to obtain the feature information of each frame better as follows:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x+y)^2/2\sigma^2} \tag{1}$$

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2}$$

where $G(x,y,\sigma)$ is a two-dimensional Gaussian kernel function that represents convolution operation, x and y are the positions of the image, σ is the smoothness coefficient of the image, $L(x,y,\sigma)$ is the Gaussian convolution of the original image at variable scale, $*$ represents convolution operation, and $I(x,y)$ is the XY coordinates of the original image.

To effectively detect stable key points in scale space, we transform them into Gaussian-difference-scale (DOG) space for feature acquisition:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,\varphi\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,\varphi\sigma) - L(x,y,\sigma) \end{aligned} \tag{3}$$

where φ is a constant of the space multiple of adjacent scales. $D(x,y,\sigma)$ can replace the Gauss–Laplace function $\sigma^2\nabla^2 G$ to establish the background.

Then, we detect extremum points of the DOG scale spatial to find the position of the feature points. Each sampling point should be compared with all adjacent points to discover whether it is larger or smaller than adjacent points in the scale domain. After that, we distribute direction parameters for feature points and determine a SIFT feature area from these three key factors: real position, direction, and scale. During the process, the directions are distributed as follows:

$$m(x,y) = \sqrt{((L(x+1),y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{4}$$

$$\theta(x,y) = \dot{g}\tan 2\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{5}$$

where m $(x,\ y)$ means length, $\theta(x,y)$ means angle, and $\dot{g}$ means the larger value of the Gaussian difference operator.

So far, the feature points of the image have been detected, and each feature point has three pieces of information: location, scale vector, and direction so that a SIFT feature area can be determined. To match the feature points between frames, we calculate the Hamming distance $D(V_p, V_q)$ and feature vector between each feature point in the current standard frame and the shaking frame, where V_p is the feature vector of a feature point P in the shaking frame, and V_q is the feature vector of the nearest feature point Q in the first frame. The smaller $D(V_p, V_q)$ is, the more similar the two features are.

Finally, we use the linear affine transformation matrix from the designed image to the first frame image to match pairs. The linear affine transformation matrix is defined as follows:

$$[A|b] = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ -a_{12} & a_{11} & b_2 \end{bmatrix}. \tag{6}$$

Then, the shaking frame can be a regression as follows:

$$\left[\hat{A}|\hat{b}\right] = \arg\min_{[A|b]} \sum_i u[i] - Av[i]^r - b \tag{7}$$

Table 1 Main notations used in this paper

Notations	Definitions
x	Horizontal coordinates in pixels
y	Vertical coordinates in pixels
σ	Smoothness coefficient of the image
$G(x, y, \sigma)$	Two-dimensional Gaussian kernel function
$L(x, y, \sigma)$	Gaussian convolution of the original image at variable scale
$I(x, y)$	XY coordinates of the original image
φ	Constant of the space multiple adjacent scales
$D(x, y, \sigma)$	Difference of Gaussian scale-space
$m(x, y)$	Length in pixels
$\theta(x, y)$	Angle in pixels
$\dot{g}$	Larger value of Gaussian difference operator
$[A\|b]$	Linear affine transformation matrix
a_{mn}	Coefficient of variation in the linear affine transformation matrix
$[\hat{A}\|\hat{b}]$	Affine regression matrix
$u[i]$	Feature point i in the shaking frame
$v[i]$	Feature point i in the standard frame
A, B	Structural element
$p(x)$	Probability density function of the k th Gauss model
$\omega_{k,t}$	Weight of the k th Gauss model in t th frame
α	Square of the learning rate
α_{ct}	Updated learning rate
π_k	Weight of the k th Gauss model
t_i	Basic model establishing time
c	Gap time
$\overline{z}$	Input data
$\overline{w}^T$	Parameter matrix of regression model
$\overline{k}^{xx'}$	Kernel matrix
$\boldsymbol{\sigma}^2$	Coefficient of the Gauss kernel matrix
F(x)	Fourier transform matrix
$\hat{f}(\overline{z})$	Possible vehicle position between different frames
$\hat{a}$	Kernel regressor
a_n	New tracking learning kernel model
a'	Prediction model of detection results
$\overline{A_{G,1}}$	Real new position
m_0	Learning rate of tracking position update
β	Conditional variable
$\overline{A_{G,0}}$	Detection ROI results of the GMM method
$\overline{A_{k,0}}$	Tracking ROI of the KCF tracker
$corx/cory$	Left/right angular coordinates
$\overline{P_{(i,k)}}$	Real point of the k th vehicle in the i th frame
$\beta_1, \beta_2, \beta_3, \beta_4$	Position control parameters
L_r	Designing real trajectory
ΔK	Trajectory coincidence matrix
C_1, C_2, C_3	Trajectory control parameters

where $u[i]$ is the feature point i in the shaking frame, and $v[i]$ is the feature point i in the standard frame.

Since the positional relationship between frames is linear, we modify the current frame through the affine matrix $\left[\hat{A}|\hat{b}\right]$ to obtain the fixed coordinate position of the dithered frame.

Through the above methods, we match the feature points between different video frames and correct the coordinate difference of the overall position between frames using the linear change matrix. We could finally eliminate the interference of jitter in UAV video on detection and improve the robustness of the general framework.

3.1.2 Connected region area operation

To give better play to the detection characteristics of the GKB framework, we transform the background into the binary (or multi-valued) image using the threshold segmentation. We extract the feature targets of different regions by calibrating the feature points of connected regions and preprocessing with opening and closing operations.

Firstly, we use the open operation to smooth the contour of the vehicles, disconnect the narrow discontinuity and eliminate the prominence as follows:

$$A \circ B = (A \ominus B) \oplus B \tag{8}$$

where $\circ$ means open operation, $\ominus$ means corrode operation, and $\oplus$ means expand the operation.

To open A with structural element B is to corrode A with B, then expand the result with B. In patents, convolution kernels (B) with decreasing pixels are used to open the original binary image (A) to remove the outliers outside the target vehicle. The video is corrupted with the background expanding. But the corroded area will shrink one circle. Therefore, an operation is needed to expand the target (vehicle) area to its original size.

So, we use the close operation as follows:

$$A \cdot B = (A \oplus B) \ominus B \tag{9}$$

where means close operation, $\ominus$ means corrode operation, and $\oplus$ means expand the operation.

To close A with structural element B is to expand set A with B, and then erode the result with B. In patents, convolution kernels (B) with decreasing pixels are used to open the original binary image (A) to eliminate the disorder and cracks from the open operation. The faults in the target vehicle are eliminated. And the edges of the vehicle are smoothed. But the target vehicle area also expands one circle outward. The corrosion operation is needed to restore the vehicle area in the image to the previous size.

Considering the small target pixel in UAV video, we use the third-order cycle iteration of open-close operation to deal with the problems such as the closely spaced vehicles. Using gradient decrement of the convolution core, we process the original image from coarse to fine, making the edge sharpness of the final processing more accurate. Because iterative processing erodes the edges of open operations locally, it increases the anti-interference ability of recognition methods for possible transversal and disturbance of video (Fig. 1).

At last, with the hole filling operation, for the 0-value noise appearing in some local open operations, 1 value is added according to the surrounding connected region. The vehicle position is reproduced and clearer.

3.2 Detection and tracing

To improve the detect efficiency in complex and changeable scenes, a novel framework is proposed to detect and trace the trace in UAV videos efficiently. There are two main parts to this part. One is optimizing the detection and tracking algorithm framework to improve overall aerial video efficiency and adaptability. The other is to establish a bidirectional feedback mechanism in the tracking process. The processing mechanism is shown in Fig. 2 as follows:

3.2.1 Optimized gauss mixture model

In aerial video analysis, the detection of moving objects is the focus of the problem. We adopt the optimized Gauss mixture model to obtain more accurate detection results to extract the moving foreground from the current frame. Among the similar algorithms, updating the background by the weighted average of the current frame and background frame in the video is more reliable and robust. Besides, the OGMM method is much better suited for our GKB model with extensive processing capacity, high stability, and high flexibility.

The essence of GMM is to fuse several single-Gaussian models to make the model more complex and produce more complex samples. For an input video Z^*, assuming the GMM is composed of K Gaussian models, in the OGMM model converted from video images, the probability density function is shown as follows [29]:

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k) = \sum_{k=1}^{K} \pi_k N(x|u_k, \sum k) \tag{10}$$

$$\sum_{k=1}^{K} \pi_k = 1 \tag{11}$$

where $p(x|k) = N(x|u_k, \sum k)$ is the probability density function of the k th Gauss model, which can be seen as the probability of generating x when the k th model is selected.

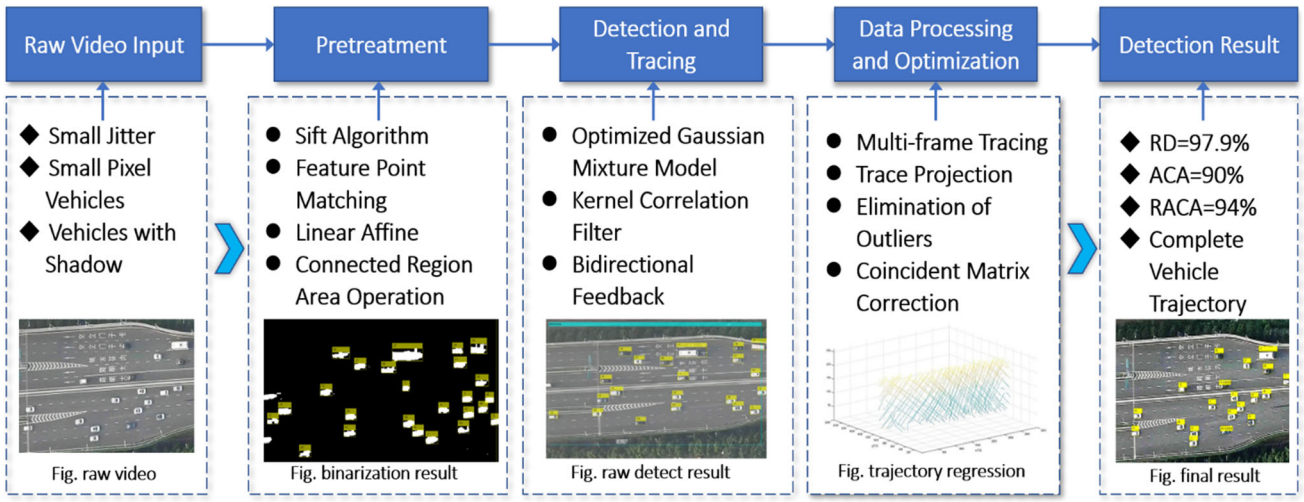

Fig. 1 Schematic diagram of the algorithm framework

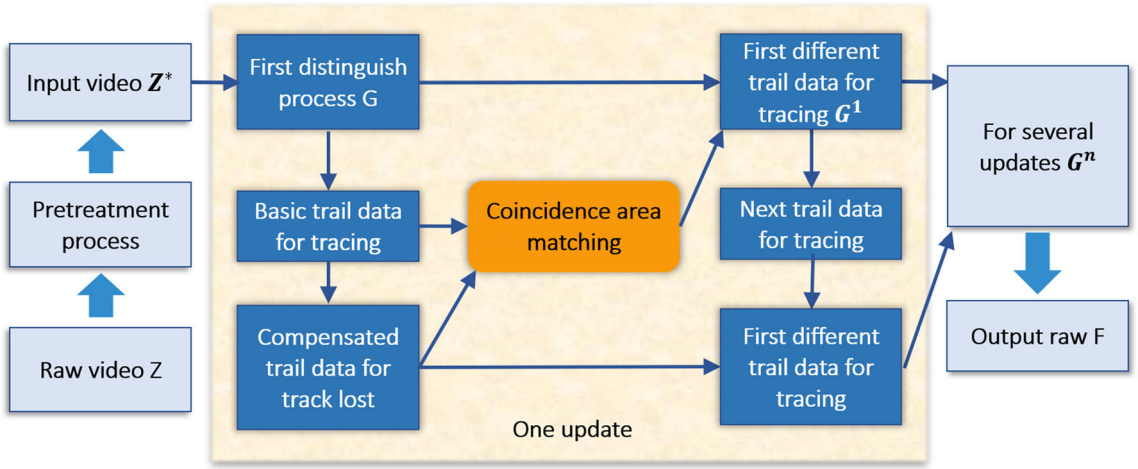

Fig. 2 Mechanism of the GKB method

$p(k) = \pi_k$ is the weight of the k th Gauss model, also called prior probability.

The weight of the different models is continuously updated when the video is playing. To decrease the computational effort, we propose a new weight updating method as follows:

$$\begin{cases} \omega_{k,t} = (1-\alpha)\omega_{k,t} + \alpha(M_{k,t}), & t < t_i \\ \omega_{k,ct} = (1-\alpha_{ct})\omega_{k,ct} + \alpha_{ct}(M_{k,ct}), & t \geq t_i \end{cases} \tag{12}$$

$$\alpha_{ct} = \alpha / \ln(t_i) \tag{13}$$

where $\omega_{k,t}$ means the weight of the k th Gauss model in t th frame, α is the square of the learning rate, α_{ct} is the updated learning rate, $M_{k,ct}$ is 1 for the matching component, and for other components, it is set to 0, t_i is the basic model establishing time, and c is the gap time.

This is a gradient decreasing updating mode, which is exceptionally suited for the GKB framework. First of all, after the pretreatment of the UAV video, the video a relatively stable so that this model can provide the OGMM efficiency to the greatest extent. Secondly, it can significantly reduce the filter interference and ensure the model stability in the long-time video detection. Thirdly, considering the only change of the α can be offset in the expectation maximization (EM) solving model, the change is compensated in the M-step as follows. So, the computed amount in the solving step will not increase.

After that, the background model is relatively stable. And the moving vehicles can be successfully detected in the UAV video.

3.2.2 Kernel correlation filter and GKB mechanism

Considering the accurate extraction of small pixel targets in different scenes, a novel Kernel correlation filter (KCF) method is proposed based on the OGMM results. The core

idea of the KCF algorithm is to expand the number of negative samples to enhance the performance of the tracker. In our framework, the core part of sample acquisition is by the cyclic matrix and ROI from OGMM. Besides, the new sample images can also be obtained by moving different pixels upward and downward, respectively, which can significantly expand the sample library to get better tracking results.

Specifically, the OGMM model results are the input of the KCF method, and the OGMM would find the actual position $\left(X_{t,i}, Y_{t,i}\right)$ for each vehicle in each frame from the whole scene. According to our regression model, we can get:

$$f(\overline{z}) = \overline{w}^T \overline{z} \tag{14}$$

where $\overline{z}$ is the input position data, $\overline{w}^T$ is the parameter matrix of the regression model (regression), $f(\overline{z})$ is the result of regression, which also can be called the response value.

We use the least square regression and take the regression function of $\overline{w}$ as followed [30]:

$$\overline{w} = \left(X^H X + \lambda I\right)^{-1} X^H \overline{y} \tag{15}$$

where the parameters of X matrix correspond to the sample value, the parameters of $\overline{y}$ matrix corresponds to the regression value.

The ridge regression function is the most significant part of the KCF, it directly influent the tracking results. In the framework, we use the kernel function and regression to solve the function as follows:

$$\bar{k}^{\overline{x}x'} = \exp\left(-\frac{1}{\sigma^2}\left(\|\bar{x}\|^2 + \|\overline{x'}\|^2 - 2F^{-1}\left(\sum_c \hat{x}_c^* \odot \hat{x}_c^{'}\right)\right)\right) \tag{16}$$

where $\overline{k}^{xx'}$ is kernel matrix, $\frac{1}{\sigma^2}$ is the coefficient of the Gauss kernel matrix, $F(x)$ is the Fourier transform matrix, $\hat{x}$ is the cyclic matrix of the $\overline{x}$, and $\odot$ is the element-wise product operation in the Fourier transformation.

So, a nonlinear regressor $\hat{a}$ can be trained for our input aerial video with the HOG feature graph:

$$\hat{a} = \frac{\hat{y}}{\widehat{k^{xx}} + \lambda}. \tag{17}$$

In the next frame, the vehicle image is selected in the target position frame of the previous frame, and the HOG feature map is obtained by cosine weighting. Then, by computing of response matrix in the Fourier domain, we can get the response matrix $\hat{f}(\overline{z})$ for the possible vehicle position between different frames as follows:

$$\hat{f}(\overline{z}) = \widehat{k^{\overline{x}z}} \odot \hat{\alpha}. \tag{18}$$

So that we can find the maximum response position in matrix $f(\overline{z})$, if the response value exceeds the given cosine threshold, then the position is the positions of vehicles in the current frame; if the maximum response value is still less than the threshold, the following positions need to be remedied. That is one reason the KCF track result would appear to track the lost phenomenon. At that time, The OGMM helped output the detect positions of the vehicles to join the matrix. When the maximum response value is less than the threshold, the KCF tracker can find the nearest detection position as the value to judge again. Most of the time, the detection result can successfully make up the track lost phenomenon.

Finally, the KCF tracker updates the model. To find the accurate contour, the model repeats the steps as before by selecting the sample at the newly found vehicle's location, calculating the model for the current frame, and marking it. The model for the next frame is calculated by linear interpolation as follows:

$$a_n = m_0 a_o + (1 - m_0) a' \tag{19}$$

where a_n is new model, m_0 is the learning rate, a_o is the tracking prediction model, and a' is the prediction model of detection results.

Using this kind of learning mode, the KCF tracker can find the whole trajectories of the vehicles in the detection area, which are also the real positions of the vehicles. Moreover, its region of interest (ROI) area is more accurate than the OGMM result because the tracing model is more robust in light of weather conditions. So that the real position areas renew as follows:

$$\overline{A_{G,1}} = \frac{\beta}{1+\beta}\overline{A_{G,0}} + \frac{1}{1+\beta}\overline{A_{k,0}} \tag{20}$$

where β is a conditional variable, $\overline{A_{G,0}}$ is the detection ROI results of the GMM method, and $\overline{A_{k,0}}$ is the tracking ROI of the KCF tracker.

After the data quality control, the new $\overline{A_{G,1}}$ would join into the following KCF tracking process as the input of the vehicle positions and the compensation position (see Fig. 2). For several times of renewing processing until the training results attain the threshold, a high-precision vehicle position result F can be obtained.

3.3 Data processing and optimization

We need to perform the data cleaning and optimization steps to obtain more accurate trajectory coordinate data and vehicle contour position. Firstly, through the vehicle's dynamic characteristics and driving speed, we carry out trajectory regression and initial data cleaning according to the identification coordinates obtained by OGMM results.

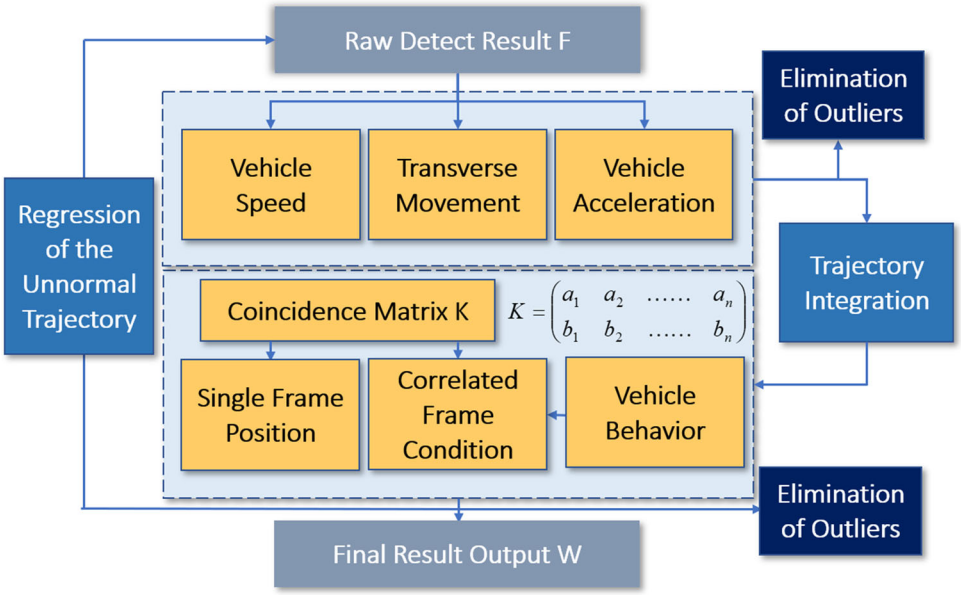

Fig. 3 Mechanism of the data processing

Secondly, using the coincidence matrix K, we match the regression results with the KCF tracking position to get a relatively accurate designing real trajectory L_r. At last, we eliminate the abnormal points, mainly caused by 'ghost ROI' or false detection, and correct the vehicle contour through the correlation control function to output the final high-precision result set W (as shown in Fig. 3).

In the beginning, to get better results, we need to define and analyze the real positions and the characteristic of the vehicles by the correlated-frame trajectory and extract the trajectory of the vehicles from the OGMM results.

Define the center of the frame selection matrix as the real point of the k th vehicle $\overline{P_{(i,k)}}$ in the i th frame:

$$\overline{P_{(i,k)}} = ((corx1 + corx2)/2, (cory1 + cory2)/2) \tag{21}$$

where $corx$ and $cory$ is the left/right angular coordinates.

Considering the behaviors of the vehicles, the vehicles have the characteristics such as the speed range, transverse movement range, and acceleration range. Combined with these characteristics, we can arrange the trajectory of the vehicles by the following:

$$\overline{P_{(i+1,k)}} = \min_j(dis(\overline{P_{(i,k)}}, \overline{P_{(i+1,j+1)}}). \tag{22}$$

The $\overline{P_{(i+1,k)}}$ need to meet the conditions of the following characteristic of the vehicles, including:

Speed control:

$$dis\left(\overline{P_{(i,k)}}, \overline{P_{(i+1,k)}}\right) < \beta_1. \tag{23}$$

Transverse movement(in the expressway) control:

$$\beta_2 < angle\left(\overline{P_{(i,k)}} —— \overline{P_{(i+1,k)}} —— \overline{P_{(i+2,k)}}\right) < 180°. \tag{24}$$

Acceleration control:

$$\beta_3 < \frac{dis\left(\overline{P_{(i,k)}}, \overline{P_{(i-1,k)}}\right)}{dis\left(\overline{P_{(i,k)}}, \overline{P_{(i+1,k)}}\right)} < \beta_4 \tag{25}$$

where β_1, β_2, β_3, β_4 are the control parameters.

From the vehicles' behavior performance in the pixels, we can quickly eliminate the abnormal data and regress the raw trajectories of the vehicles. However, these raw trajectories come from the raw position data, so their reliability and accuracy must be tested and optimized.

According to these two sets of vehicle trajectory data, we judge their coincidence degree and generate the coincidence matrix K of trajectory.

Definition trajectory results are as follows:

$$\begin{cases} L_1 = |a_1 \quad a_2 \quad \cdots \quad a_n| \\ L_2 = |b_1 \quad b_2 \quad \cdots \quad b_n| \\ L_r = \dfrac{L_1 + L_2}{2} \end{cases} \tag{26}$$

where a_n is the result of the trajectory regression of OGMM detection, b_n is the result of the trajectory of the vehicles from the KCF method, and L_r is the real designing trajectory.

Definition coincidence matrix ΔK is as follows:

$$\Delta K = \begin{vmatrix} \Delta a_1 & \Delta a_2 & \cdots & \Delta a_n \\ \Delta b_1 & \Delta b_2 & \cdots & \Delta b_n \end{vmatrix} \tag{27}$$

where the Δa_n and the Δb_n are the designing deviation, which are the different values from the OGMM and KCF results to the real designing position. The ΔK matrix is the integration results of the GKB algorithm, from which we can judge whether a designing trajectory L_r is reliable.

For a specific ΔK matrix, we have the following judgment conditions:

$$\begin{cases} \sum_i a_i < C_1 \\ \sum_i b_i < C_1 \\ a_i < C_2 \\ b_i < C_2 \\ \sum_i (|a_i - b_i|) < C_3 \end{cases} \tag{28}$$

Firstly, the value of the $\sum a_i$ and $\sum b_i$ should be less than the limited trajectory mistaken value C_1, which could be recorded as a reliable trajectory. Secondly, we judge each vehicle's position a_i and b_i. If they are less than the limited single mistaken value C_2, they are recorded as reliable coordinates. Thirdly, we judge the GKB method difference $\sum_i (|a_i - b_i|)$, which should be less than the limited method mistaken value C_3. Generally speaking, most ΔK matrices have completely reliable trajectories and coordinates. For these parts of ΔK matrics, the data in it are smoothed to form a real trajectory. For unreliable coordinates, the tracking trajectory is judged by three parts in the vehicle identification trajectory regression process. After eliminating the outliers, the new ΔK matrix is judged repeatedly. Until the position accuracy attains the calibrated threshold, the final vehicle positions in L_r is the real detection positions of our whole framework.

3.4 Experiment design

In the data collection process, the research team collected two typical freeway sections in Nanjing, China, by the unmanned aerial vehicle (DJI Mavic professional). The videos contain specific traffic behaviors such as congestion formation, merging and diversion in weaving areas, and traffic conflict, which are of value in theoretical research. Besides, they are different in shoot condition, height, weather condition, road geometry, and traffic condition, which is better to verify the effectiveness and robustness of our GKB framework.

Table 2 Information on the UAV video shooting conditions

Information	Video #1	Video #2
Road geometry	Straight section	Curve section
Frame rate	25fps	25fps
Resolution	3860*2160	3860*2160
Weather condition	Sunny	Cloudy
Shoot condition	Small jitter	Shaking
Duration	18 s	27 s
Height	207 m	230 m
Traffic condition	Free flow	Congestion flow

The detailed shooting conditions are shown in Table 2. Video #1 is shot in a straight section on a sunny day, so it has a relatively apparent phenomenon of vehicle shadow. Video #2 is shot in a curve section on a cloudy day, so it has a shaking phenomenon. The traffic conditions of the two videos are free flow and congestion flow. The UAV flying heights for the two videos are 207 m and 230 m. And the length of the two videos is 569 frames and 803 frames. The video resolution is 3840*2160. The frame rate is 25 fps. The test environment is based on the platform of MATLAB 2016b. The test equipment is based on a memory laptop with Inter Core I5-8300HQ@3.6 Hz CPU processor, 4G system memory, and GeForce GTX 950 graphics card. We believe that the running speed of the GKB framework can be significantly improved with better equipment.

As an unsupervised machine learning framework mode, the most significant advantage of the GKB framework is that the algorithm can maintain its strong robustness without pre-training and adapt to different scene environments through as few parameter changes as possible. In our method, the parameters of most contents are relatively opposite, and there is less interference and influence on each other.

According to the specific needs of the experiment, a set of recommended parameters is given as follows: (i). Preprocessing part: The main influencing factor of parameter selection is the average pixel size of the target of interest. Under the shooting accuracy of 4 K, when the UAV flying altitude is lower than 200 m, the vehicle pixel size is nearly 50 * 24 (pi)—200 * 65 (pi). According to the experimental data, the best-recommended parameter of the convolution kernel of the opening and closing algorithm is M = (27,9) to m = (9,9). When the UAV's flying altitude is higher than 200 m, the vehicle's pixels are often below 50 * 24 (pi), so the best-recommended parameter is M = (12,3) to m = (3,3). (ii). Detection and tracking part: According to the duration of the videos, the test modeling process of A = 100 frames is selected, and the learning rate α = 0.005, which is the main factor affecting the accuracy and speed of the model. The basic model establishes time t_i = 40 to get background, the gap time c = 10 frames is better to establish and update the background. During the process of the bidirectional feedback, the learning rate m_0 = 0.75, the conditional parameter $\beta = 0.55$, which are the main factors affecting the accuracy of vehicle trajectory in different scenes. (iii). Data processing part: According to the performance of essential vehicle dynamics and current driving speed, the control parameters in the correlated-frame trajectory process are $\beta_1 = 20(pixel)$, $\beta_2 = 130°$, $\beta_3 = 4.45$, $\beta_4 = 8.5$, which are the critical control parameters to eliminate outliers. For the coincidence matrix ΔK, according to the shooting accuracy and height, the optimal

vehicle contour control parameters are $C_1 = 95(pixel)$, $C_2 = 7.5(pixel)$, $C_3 = 55(pixel)$. Such parameter settings can be helpful for eliminating the system's error and significantly improve the accuracy of the detection results.

4 Experiment results

4.1 Measurement of goodness of fit

The results come from the UAV video by the GKB algorithm, including the real positions and the contours of the vehicles, every detection vehicle picture in every frame, the regression trajectory, and the residual from the ΔK matrix. To significantly validate the performance of GKB results, we still define other indexes to help us better judge the data's reliability.

The algorithm effect time:

$$\mathrm{AT} = \frac{\text{real time}(s) \text{ in detect 25 frame}}{1s}. \tag{29}$$

The rate of the successful detection:

$$RD = \frac{\sum_{i,j} \text{detect number of vehicles}}{\sum_{i,j} \text{real number of vehicles}} \tag{30}$$

where i means in the i th frame, and j means the j th vehicle in detection.

The rate of the successful detection of the dark vehicle:

Dim the dark vehicles as the average pixel value is less than γ_1, and in the test,$\gamma_1 = 25$ (including black, dark blue, red, dark brown, etc.).

$$DRD = \frac{\sum_{i,j} \text{detect number of dark vehicles}}{\sum_{i,j} \text{real number of dark vehicles}}. \tag{31}$$

The rate of the successful detection of the closely spaced vehicles:

Dim the closely spaced vehicles as the two ROI borders nearby pixel position is less than γ_2, and in the test,γ_2 = 12 (12 is the enormous convolution core in the open–close algorithm)

$$NRD = \frac{\sum_{i,j} \text{detect number of closely} - \text{spaced vehicles}}{\sum_{i,j} \text{real number of closely} - \text{spaced vehicles}}. \tag{32}$$

The rate of the reliability of the ROI area:

$$\mathrm{RCA} = \frac{\text{coincidente area}}{\text{detect area}}. \tag{33}$$

The average rate of the whole detection ROI reliability:

$$\mathrm{ACA} = \mathrm{avg}_{(i,j)}\left(\frac{\text{coincidente area}}{\text{detect area}}\right) = \mathrm{avg}_{(i,j)}(RCA). \tag{34}$$

The rate of 'perfect' detection vehicles:

$$RACA = \frac{\sum_{i,j}(RCA > \gamma_3)}{\sum_{i,j} \text{real number of vehicles}}. \tag{35}$$

In the experience test, to get a high-precision vehicle position, we set the parameter γ_3 = 0.85.

The index to determine whether the result is reliable mainly depends on the RD, ACA, and RACA, which closer to 1 is better.

4.2 Outputs of methodological procedures

The outputs from each step in our framework are shown in detail to guide our research's functions. The pretreatment examples are shown in Fig. 4b, c. It can be seen that before the pretreatment, the background after binarization is seriously interfered by the non-vehicle factors such as lane lines and guideposts. Moreover, the ghost areas appearing, such as in frame #40 and #110, can greatly influence the final result. After the pretreatment, the background becomes much clearer, and the foreground of the vehicle is easier to detect and track (see Fig. 4).

The GKB method examples are shown in Fig. 4d, e, f. Figure 4.d shows the detection result of the single detection model using GMM, which has the problems of the detection loss of the adjacent vehicles and dark vehicles, ghost areas, and low ACA and RACA. Figure 4e shows the result of the unidirectional track after detection method using GMM detection and KCF tracking, which have the lost track and low RACA problems. And Fig. 4f shows the result of our GKB algorithm, which overcomes the above issues and superiorly detects all vehicles in the UAV video.

The data processing examples are shown in Fig. 5. It can be seen that some of the vehicle positions are in low RCA conditions by different causes, including transverse interference (TI), vertical interference (VI), road interference (RI), and abnormal detection (AD), which significantly influence the data mining and R&D work. These raw results always have the bad Δa_i and Δb_i and very volatile. After the data cleaning method, nearly all the vehicle positions with RCA, about 70–80% could be repaired into a 'perfect' position. More than half of the positions with RCA, about 60–70% could be fixed into a 'perfect' position. For these abnormal positions (RCA < 0.6), the outliers could be eliminated or fixed into regular positions.

4.3 Results of vehicle detection using GKB framework

To present the method results more intuitively and better help us test the impact of the experience, we mark the real positions and areas of each vehicle in about one hundred

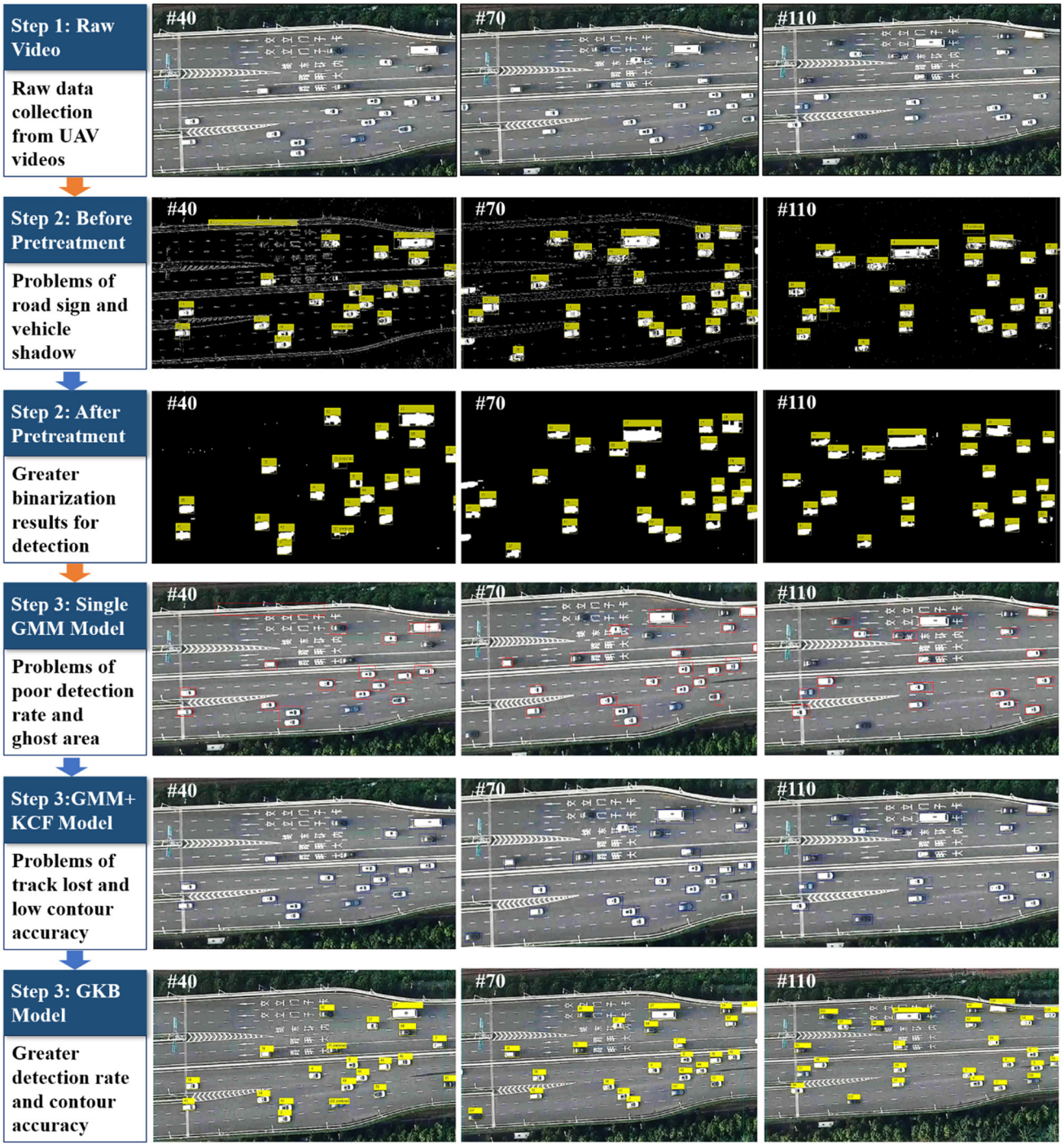

Fig. 4 Comparing results with different framework in video #1 frame 40, 70, 110 **a** raw video; **b** binarization results before pretreatment; **c** binarization results after pretreatment; **d** detect results using the single Gaussian mixture model; **e** detect result using the track-before-detect (GMM + KCF) model; **f** detect result using GKB model

frames of each video to compare and get the rate results. Besides, we also tested video in various other methods, including single GMM, tracking-after-detection, and one-stage deep learning method (YoloV4 with default weight). The results are as follows:

The result indexes of different methods in video #1 are shown in Table 3. In Table 3, we collect the essential test parameter of detecting vehicles and the algorithm effect time in the video. Then, we calculate the main depending indexes of the test RD, DRD, NRD, ACA, and RACA compared with the real marked

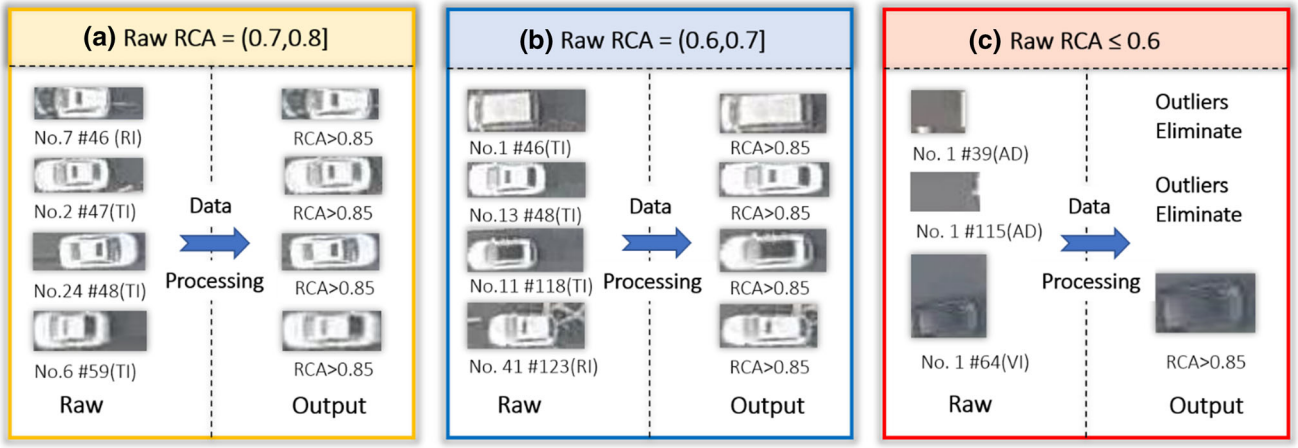

Fig. 5 Data processing results with difference RCA **a** the processing of the vehicles whose RCA are between 0.7 and 0.8; **b** the processing of the vehicles whose RCA are between 0.6 and 0.7; **c** the processing of the vehicles whose RCA are below 0.6

Table 3 Results index of the different frameworks using in video #1

Index of results in video #1	Single GMM detect	Unidirectional track after detect result	GKB result(our)	YoloV4(darknet)
Number of vehicles been detected	74(whole video)	93(whole video)	**99(whole video)**	87(whole video)
Algorithm effect time(s)	16.5(whole video)	24.0(whole video)	4.5(whole video)	**2.1**(whole video)
Number of RCA < 0.6 (Times, frames)	25	9	**1**	6
Number of RCA < 0.7 (Times, frames)	58	25	**4**	11
Number of RCA < 0.8 (Times, frames)	139	78	**10**	65
Number of RCA < 0.85 (Times, frames)	201	115	**19**	119
RD(%)	86.1	94.4	**97.9**	92.9
DRD(%)	45.0	62.4	**82.1**	67.1
NRD(%)	37.9	65.9	**90.3**	75.6
ACA(%)	74.2	79.8	**90.1**	80.3
RACA(%)	72.4	81.5	**93.8**	81.1

Bold indicates the best performance in the corresponding index

positions of the vehicle we picked up. From the results, under relatively stable video conditions, it is evident that the single GMM model is relatively poor in the location and accuracy of the detection. And unidirectional feedback method (GMM + KCF method) merely gets a reasonable detection rate. At the same time, although the Yolov4 method has a higher speed and maintains relatively good detection and position accuracy, its detection ability for dark vehicles and adjacent vehicles is still insufficient. Only by the GKB framework can we receive a set of high-precision vehicle detection results, especially for accurately controlling vehicles' contour.

The proposed framework is also used in video #2 where the road geometry is curved, the flight height of the UAV is much higher, the video is slightly shaking, the traffic conditions are more complex, and the number of vehicles is several times than video #1. The detection results are shown in Fig. 6, including the whole detection of video #2 (the number with predicted means that the vehicle's position is the feedback result), detection examples of special vehicles, and the processing of trajectory regression. The result indexes are shown in Table 4. The traditional detection method is much weaker in complex conditions and congestion traffic flow. At the same time, the Yolov4 algorithm also performs poorly in this complicated scene, and its overall recognition efficiency is not as good as the track-after-detect method. The GKB framework detection rate is nearly twice that of dark and closely spaced vehicles compared to the single model. Besides, our algorithm can still collect the relatively accurate vehicles' contour, nearly 20% over the YoloV4 method, even on the curve road.

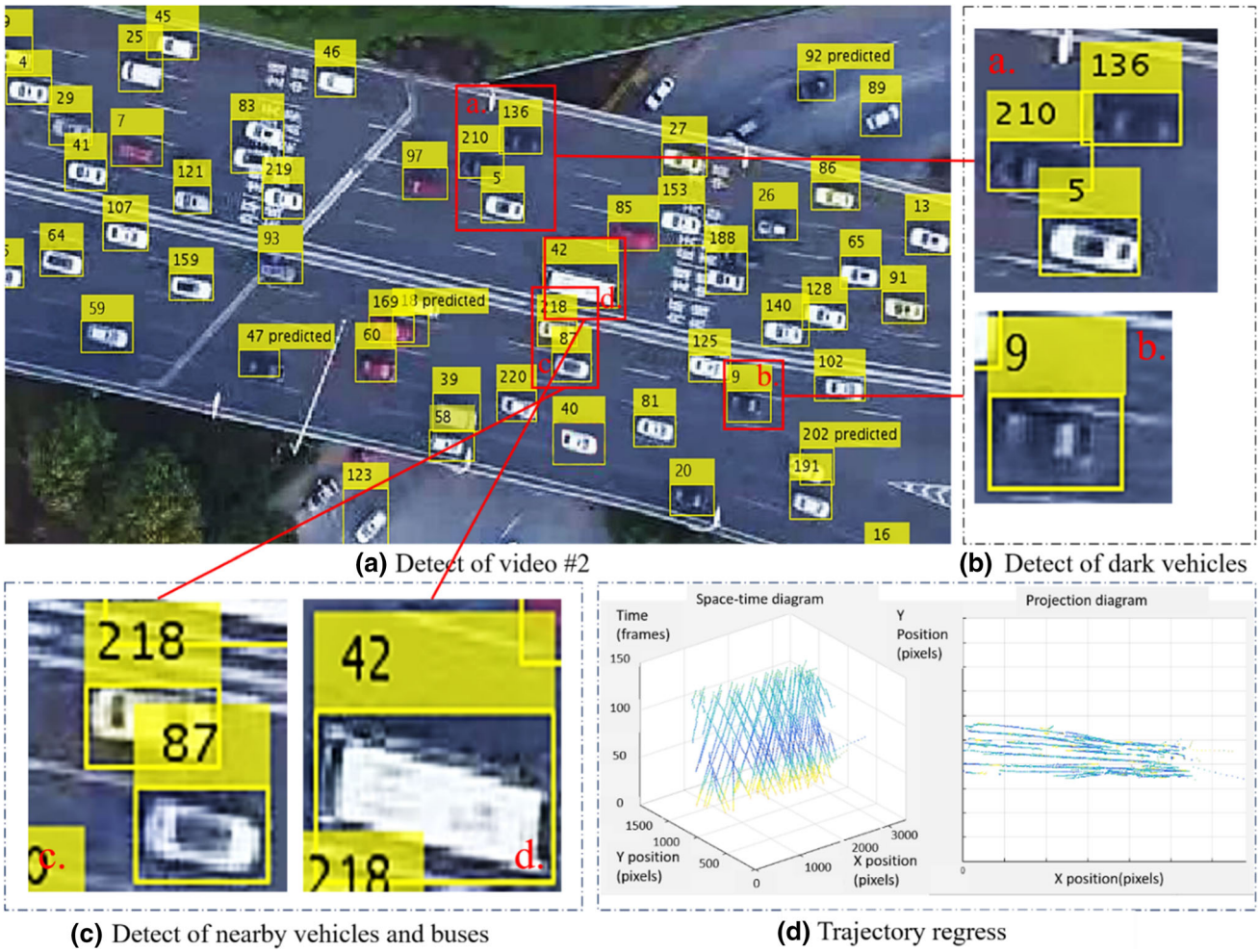

Fig. 6 Detection results of the GKB framework in video #2 **a** detection results in video #2; **b** detection results of dark vehicles; **c** detection results of nearby vehicles and buses; **d** results of trajectory regression

Table 4 Results index of the different frameworks used in video #2

Index of results in video #2	Single GMM detect	Unidirectional track after the detect result	GKB result	YoloV4(darknet)
Number of vehicles been detected	420(whole video)	638(whole video)	741(whole video)	552(whole video)
Algorithm effect time(s)	22.5(whole video)	31.5(whole video)	**5.0**(whole video)	**2.1**(whole video)
Number of RCA < 0.6 (Times, frames)	329	119	**18**	65
Number of RCA < 0.7 (Times, frames)	> 800	245	**38**	259
Number of RCA < 0.8 (Times, frames)	> 800	453	**84**	554
Number of RCA < 0.85 (Times, frames)	> 800	> 800	**159**	> 800
RD(%)	72.3	82.0	**95.7**	75.4
DRD(%)	40.9	62.8	**81.3**	58.6
NRD(%)	40.4	58.8	**85.4**	50.7
ACA(%)	64.3	76.1	**87.9**	72.5
RACA(%)	60.8	73.7	**91.7**	70.9

Bold indicates the best performance in the corresponding index

Comparing the results between video #1 and #2, the percentage of the standard positions (RCA > 0.6) is similar to the results of the video #1, considering the seven times a whole number of the vehicles, which suggests that the GKB method has the remarkable ability to control the accuracy of the detect results. It is noted that the traditional machine learning algorithms and the Yolov4 method all have a sharp decrease in RD value, mainly due to the scene's complexity, the interference of video, and the reduction in the vehicle pixel value. Even so, the GKB algorithm can obtain rather high-precision positions and trajectories of vehicles, which is extremely important for traffic analysis.

Validation of the detection performance with low illumination conditions would be helpful for practitioners. Our current study does not collect the UAV videos in very low illumination conditions such as night data. However, our proposed methodology is believed to adapt well to low illumination environments. We proposed the enhancement model for complex conditions such as closely spaced and dark vehicles with morphological algorithms and data processing. The results demonstrate the effectiveness of the detection enhancement algorithms. Thus, the algorithms are expected to have good detection performance for low illumination conditions.

In this study, the proposed methodology framework is tested and validated on two datasets in which traffic status is free-flowing and congested. The results show that the proposed models perform well for vehicle detection and trajectory construction. Though not evaluated in the present study, the proposed framework can still work well under complex and dynamic traffic conditions such as stop-and-go traffic. The reason is that the proposed methodology first detects the vehicle shape in the image and then correlates the frame detections into the vehicle trajectories according to the vehicle movements. Thus, the framework does not require traffic conditions because it does not rely on traffic flow theories. Therefore, the algorithms should have extreme reliability, adaptability, and robustness.

5 Conclusion and future work

This research proposes an enhancing precision vehicle detection GKB framework for UAV video. Firstly, we adapted SIFT feature extraction, KNN matching, and linear affine transform to eliminate the video shaking and connected region area operation to reduce the interference of external factors. Secondly, we proposed the GKB framework, which enhances detection through inter-frame position features. Besides, the framework proposes coordinate information optimization based on two-way feedback, which solves some detection problems of traditional algorithms, such as missed inspection and the 'ghost Region of Intersect (ROI)' area. Thirdly, we use correlated-frame trajectory integration and coincident matrix to promote the accuracy of the trajectory and eliminate the abnormal position. The framework has strong reliability, adaptability, and robustness, which can better detect small pixel vehicle targets in the case of changeable scenes and UAV jitter.

The results show that the proposed approach significantly improves vehicle detection. The total accuracy of our model is 98%, which is a 11% improvement over the traditional single detect model and a 4% improvement over the track-after-detect method. Our model's detection rate of closely spaced and dark vehicles is improved by 15–25% compared to previous methods. Our model's vehicle contour detection accuracy is over 94%, which is about a 15% improvement over previous methods.

Our study provides a more straightforward way for traffic researchers to obtain vehicle trajectory information in each frame for each vehicle from UAV video, including the position, horizontal and vertical location in the road coordinate, speed and acceleration, and so on, which can better help traffic researchers in traffic flow modeling, traffic congestion analysis, and traffic conflict evaluation. At the same time, using our method, we can quickly collect the high-precision training database required for deep learning research with different scenes. The UAV video data is crucial for other researchers to validate and compare their models. We have decided to publish the data for researchers to access on the website www.seutraffic.com.

In the future, we plan to expand our research scale and conditions. Firstly, we will further improve the adaptability of the algorithm to detect more traffic elements, including pedestrians, non-motor vehicles, and other motor vehicles on urban roads. Secondly, we will further enhance the robustness of our framework under the conditions of low resolution, severe shadows, and shadow interference. Thirdly, we will combine our method with the advanced algorithm and expand our traffic database for future research.

Data availability The UAV video and vehicle trajectory data that support the findings of this study are available for access on the website http://www.seutraffic.com/#/download.

Declarations

Conflict of interest The authors have no conflicts of interest to declare relevant to this article's content.

References

1. Wang Z (2013) Visual traffic jam analysis based on trajectory data. IEEE Trans Visual Comput Graph 19(12):2159–2168
2. Sopasakis A, Katsoulakis M (2016) Information metrics for improved traffic model fidelity through sensitivity analysis and data assimilation. Trans Res Part B: Methodol 86:1–18
3. V-os J, Farah H, Hagenzieker M (2021) Speed behaviour upon approaching freeway curves. Accid Anal Prev 159:106276
4. Monteil J et al (2014) Calibration, estimation, and sampling issues of car-following parameters. Trans Res Record: J Trans Res Board 2422:131–140
5. Guo H, Wang Z, Yu B, Zhao H, Yuan X (2011) Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. In Proc IEEE PacificVis 20:163–170
6. AT Palma, V Bogorny, B Kuijpers, LO Alvares (2008) A clusteringbased approach for discovering interesting places in trajectories. In: Proc ACM symposium on applied computing, p 863–868
7. Lars Sommer, Nicole Schmidt, Arne Schumann1 Jurgen Beyerer (2018)Search area reduction fast-RCNN for fast vehicle detection in large aerial imagery. In: 2018 25th IEEE international conference on image processing (ICIP), p 3054–3058
8. Qiwei Peng,Wang Luo, Gongyi Hong, Min Feng, Yuan Xia, Lei YuXiaolong Hao, Xu Wang, Mingxuan Li (2016) Pedestrian detection for transformer substation based on gaussian mixture model and YOLO.2016 In: 8th international conference on intelligent human-machine systems and cybernetics, p 562–565
9. Razakarivony S, Jurie F (2016) Vehicle detection in aerial imagery: a small target detection benchmark. J Vis Commun Image Represent 34:187–203
10. Krajewski R, Moers T, Eckstein L (2019) VeGAN: using GANs for augmentation in latent space to improve the semantic segmentation of vehicles in images from an aerial perspective. IEEE Winter Conf Appl Comput Vision (WACV) 2019:1440–1448
11. Mingqiang Chen, Qingling Zhao, Zhe Jiang, Rui Xu Intrusion. Detection for in-vehicle CAN networks based on auxiliary classifier GANs In: 2021 international conference on high performance big data and intelligent systems p.186–191.
12. He K et al (2015) Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
13. Sang Jun, et al. (2017) Faster-RCNN model identification analysis. Journal of Chongqing University7
14. Medera, Dusan, and S. Babinec. (2009) Incremental learning of convolutional neural networks.In: Ijcci 2009—Proceedings of the international joint conference on computational intelligence, Funchal, Madeira, Portugal, October DBLP, p 547–550
15. Syed Tariq, Ali Kalpana Goyal. Moving object detection using self adaptive gaussian mixture model for real time applications. In: Proceeding international conference on recent innovations is signal processing and embedded systems (RISE-2017), p 27–29
16. Bouwmans T (2009) Subspace learning for background modeling: a survey. Recent Patents Comput Sci 2(3):223–234
17. Meng Liu, Chengdong Wu and Yunzhou Zhang (2007) Motion vehicle tracking based on multi-resolution optical flow and multi-scale harris corner detection In: Proceedings of the 2007 IEEE international conference on robotics and biomimetics, p 2032–2036
18. Barnich O, Van Droogenbroeck M (2011) ViBe: a universal background subtraction algorithm for video sequences. IEEE Trans Image Process 20(6):1709–1724
19. KaewTraKulPong, P., Bowden, R. (2002) An improved adaptive background mixture model for real-time tracking with shadow detection. In: P. Remagnino, G. Jones, N. Paragios, C. Regazzoni (eds.) Videobased surveillance systems, p.135–144
20. Ke R et al (2019) Real-time traffic flow parameter estimation from UAV video based on ensemble classifier and optical flow. IEEE Trans Intell Transp Syst 20(1):54–64
21. M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, (2013) The Visual Object Tracking VOT2013 challenge results. In ICCV Workshops, p 98–111
22. J.-Y. Lee and W. Yu. (2011) Visual tracking by partition-based histogram backprojection and maximum support criteria. In Proceedings of the IEEE international conference on robotics and biomimetic (ROBIO)
23. Moeslund TB, Hilton A, Kruger V (2006) A survey of advances in vision-based human motion capture and analysis. Comp Vis Image Underst 2–3:90–126
24. Ren W, Wang X, Member S, Tian J (2021) Tracking-by-counting: using network flows on crowd density maps for tracking multiple targets. IEEE Trans Image Process 30:1439–1452
25. Xinqiang Chen, Zhibin Li, Yongsheng Yang, Lei Qi, and Ruimin Ke (2019) High-resolution vehicle trajectory extraction and denoising from aerial videos. In: IEEE transactions
26. Naima Amrouche ; Ali Khenchaf ; Daoud Berkani (2017) Multiple mode multi-target tracking in high noise environment using radar measurements. In: 2017 sensor signal processing for defence conference (SSPD).
27. Z. Wang, H. Xiao, W. He, F. Wen and K. Yuan (2013) Real-Time SIFT-Based Object Recognition System, In: International conference on mechatronics and automation, Takamatsu
28. Nimbalkar AK, Kahe RR, Patil CS (2014) Patil face and hand gesture recognition using principle component analysis and kNN classifier. Int J Comput Appl 8:26–28
29. Z. Zivkovic (2004) Improved adaptive gaussian mixture model for background subtraction. In: IEEE International Conference on Pattern Recognition, p 28–31
30. C. Xie, M. Savvides, and B. Vijaya-Kumar (2005) Kernel correlation filter based redundant class-dependence feature analysis (KCFA) on FRGC2.0 data, In : analysis and modelling of faces and gestures