Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5565589, 12 pages
https://doi.org/10.1155/2021/5565589

Research Article

A Vision-Based Target Detection, Tracking, and Positioning Algorithm for Unmanned Aerial Vehicle

Xin Liu and Zhanyue Zhang

Space Engineering University, Beijing 101416, China

Correspondence should be addressed to Xin Liu; liuxin0899@sohu.com

Received 24 February 2021; Revised 15 March 2021; Accepted 23 March 2021; Published 12 April 2021

Academic Editor: Xin Liu

Unmanned aerial vehicles (UAV) play a pivotal role in the field of security owing to their flexibility, efficiency, and low cost. The realization of vehicle target detection, tracking, and positioning from the perspective of a UAV can effectively improve the efficiency of urban intelligent traffic monitoring. In this work, by fusing the target detection network, YOLO v4, with the detection-based multitarget tracking algorithm, DeepSORT, a method based on deep learning for automatic vehicle detection and tracking in urban environments, has been designed. With the aim of addressing the problem of UAV positioning a vehicle target, the state equation and measurement equation of the system have been constructed, and a particle filter based on interactive multimodel has been employed for realizing the state estimation of the maneuvering target in the nonlinear system. Results of the simulation show that the algorithm proposed in this work can detect and track vehicles automatically in urban environments. In addition, the particle filter algorithm based on an interactive multimodel significantly improves the performance of the UAV in terms of positioning the maneuvering targets, and this has good engineering application value.

1. Introduction

With the rapid increase in the number of motor vehicles in urban environments, traffic congestion, accidents, and other problems occur frequently. Under the conditions of increasingly restricted traffic conditions, intelligent transportation means must be used for improving the system efficiency and stability. Traffic monitoring is the basis for the study of various traffic problems. Traditional traffic monitoring technologies such as induction coils, geomagnetism, and roadside cameras not only have a small detection range but also have low accuracy and poor mobility, and this severely restricts the development of intelligent transportation systems [1–5]. In recent years, unmanned aerial vehicle (UAV) technology has developed rapidly. UAV is a kind of aircraft with high flexibility without manual driving. At present, UAVs are being increasingly used in areas such as postdisaster rescue, aerial photography, daily monitoring, and military observation.

The use of UAVs to detect, track, and locate vehicle targets is of great significance for the construction of smart cities. A UAV is used in diverse collection scenarios and can collect global traffic information of complex road sections, intersections, or multiple roads. In addition, it can obtain high-definition videos from the vertical upper perspective of the road and obtain key traffic parameters that cannot be extracted by conventional monitoring methods. It can adapt to diverse collection needs and can continuously monitor key areas. Therefore, it has significant advantages in traffic information collection and monitoring.

The dynamic and complex UAV videos pose severe challenges for video detection technology. According to the principles of traditional video vehicle detection, the texture of a target vehicle is used as a feature for detection. However, it is easily affected by light. The other is a textureless vehicle detection method that uses gradient calculations, and it does not perform well in complex environments and occlusion conditions. Traditional vehicle classification algorithms, such as support vector machine and classifiers that calculate the gradient histograms of the captured images as features, have the weak discriminative ability, provide unsatisfactory results, and are difficult to apply for complex and changeable

traffic roads. Therefore, it is difficult to achieve vehicle automation and precise information extraction using traditional video detection technology. In recent years, target detection methods based on the deep neural network have made significant breakthroughs in terms of their robustness and detection accuracy. The rapid development of convolutional neural network (CNN) in visual inspection has laid a strong technical foundation for the precise processing of vehicles in traffic videos captured by UAVs.

In the application scenario of a UAV aerial video, the performance of the tracking algorithm is affected by many factors such as the changes in the illumination, scale, and occlusion [6]. Unlike fixed cameras, tracking tasks in aerial videos are hampered by the low sampling rate and resolution and image jitter, which can easily lead to drift. When flying at a high altitude, the very small size of the ground objects is also a major challenge for the tracking task. In general, there are two types of video-based target tracking models, namely, the generative model and the discriminative model. The theoretical idea in the generative tracking model is that given a certain video sequence, for the target that needs to be tracked in the video, a model is built according to the tracking algorithm. Following this, the response area that is most similar to the target in the subsequent video sequence is found and is used as the target area. In this way, the tracking task continues further. The commonly used generative tracking algorithms include the optical flow method [7], the particle filter method [8], the mean-shift [9] algorithm, the continuously adaptive mean shift (CAMshift) [10] algorithm, and so on. A generative algorithm focuses mainly on tracking the characteristic information of the target itself and conducting an in-depth search on the target characteristics. However, such algorithms often ignore the influence of other factors on tracking performance. For example, severe scale changes, background information interference, or occlusion of the target can easily lead to a situation where the target cannot be tracked. The difference between the discriminative algorithm and the generative algorithm is that the former considers the influence of the background information on the target tracking task and then distinguishes the background information from the target information. In other words, to model a discriminative tracking model, it is necessary to distinguish the target and background information in a given video sequence. After the model is established, the subsequent video sequences are searched to determine further whether the searched target or background is found or not. The common discriminative tracking algorithms include correlation filtering methods and deep learning methods [11–13]. Due to the success of the deep CNN in visual recognition tasks, a large number of studies have been performed using CNN for tracking algorithms [14, 15]. These studies show that the accuracy of a CNN-based tracker is better than the tracking algorithm based on manual feature extraction.

For static targets, one can directly obtain the position and attitude information between the UAV and the target at the moment of positioning and also the angle and ranging information of the photoelectric platform into the positioning solution model, in order to perform a quick calculation of the three-dimensional coordinates of the target. However, for maneuvering targets, the platform and the target are always moving, and the tracking of the target will have interference from various factors. Under such circumstances, in order to study how a UAV can achieve target positioning with high accuracy, it is necessary to study the problem of state estimation of the UAV for a moving target. Its purpose is to use the obtained observation data to estimate the parameters such as the position and speed of the target and to estimate its current state. For nonlinear system estimation, the most commonly used filtering algorithms include the extended Kalman filter (EKF) [16], the unscented Kalman Filter (UKF) [17], and the particle filter (PF) [18, 19]. The EKF and UKF algorithms are the modified and improved forms of the linear KF algorithm, and thus, both are restricted by the KF, i.e., the system state must satisfy the Gaussian distribution. The PF algorithm is suitable for nonlinear/non-Gaussian systems and can provide good filtering effects. When the tracking target has strong mobility, the tracking performance of PF is poor, and thus, it is necessary to study the dynamic model of PF. Magill et al. [20] first proposed a multiple model (MM) algorithm that uses multiple filters to correspond to multiple target motion models. Based on the MM algorithm, Blom et al. [21] studied the interaction between multiple models in detail and proposed the interacting multiple model (IMM) algorithm with the Markov transition probability. In 2003, Boers et al. [22] proposed an IMM-based PF algorithm, namely, IMM-PF, which has a superior tracking effect for highly maneuvering targets.

In this work, vehicle detection and tracking algorithm for UAVs have been proposed based on the currently available mainstream deep learning image processing algorithms, and a vehicle target location estimation model has been designed. In particular, based on the "you only look once version 4" (YOLO v4) [23] algorithm, a vehicle detection model with superior robustness and generalization performance has been proposed. This model has been combined with the detection-based multitarget tracking (tracking-by-detection) algorithm DeepSORT [24] to realize real-time vehicle tracking. Finally, the IMM-PF algorithm has been used for achieving high-precision positioning for vehicle targets.

2. System Structure

The system structure of the automatic vehicle detection and tracking method is shown in Figure 1. A UAV uses a camera to monitor the flight area, and the acquired aerial video is transmitted back to the ground station via a data chain. At the ground station, vehicle target detection is performed on the downloaded aerial video. After the vehicle target is detected, the moving target is tracked in the subsequent video frames. In order to obtain the geodetic coordinates of the target, after extracting the pixel coordinates of the target, the latitude and longitude of the target are estimated by combining the measurement data of the position, attitude, and the camera pointing angle of the UAV, in order to realize a fully automatic detection, tracking, and positioning of the vehicle target by the UAV based on vision technology.

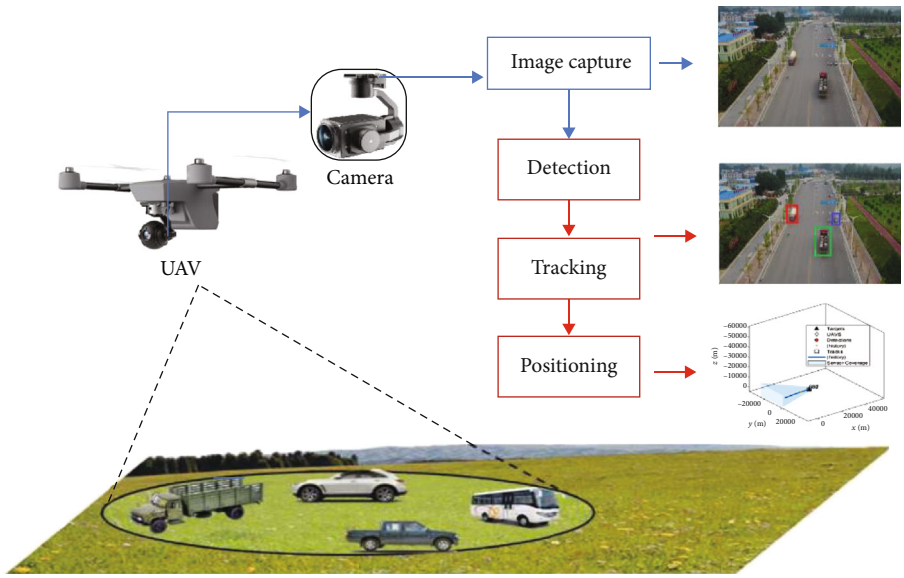


Figure 1: System structure.

3. Algorithm

3.1. YOLO v4 Target Detection Network. YOLO v4 is the latest detection network in the YOLO series, with innovations based on the integration of advanced algorithms on the basis of YOLO v3. Therefore, the YOLO v4 target detection network for vehicle detection has been used in this work. Innovations at the input of YOLO v4 include mosaic data enhancement, cross minibatch normalization (cmBN), and self-adversarial training (SAT). Innovations in the backbone network include CSPDarknet53, mish activation function, and dropblock. Innovations in the neck network involve the target detection network, often inserting a few layers between the backbone and the final output layer, such as the spatial pyramid pooling (SPP) module and the feature pyramid network (FPN) + PAN structure. The anchor frame mechanism of the prediction part of the output layer is the same as YOLO v3. The main improvement is the loss function, CIoU_Loss, during training, and the nonmaximum suppression (NMS) filtered by the prediction frame is changed to DIoU_NMS. YOLO v4 uses CSPNet and Darknet-53 as the backbone network for feature extraction. Compared to the design based on the residual neural network (ResNet), the target detection accuracy of the CSPDarknet53 model is higher, but the classification performance of ResNet is better. However, with the help of the mish activation function and other technologies, the classification accuracy of CSPDarknet53 can be improved.

In order to detect targets of different sizes, a hierarchical structure is required to enable the head of the target detection to detect the feature maps at different spatial resolutions. To enrich the input header, the bottom-up and top-down data streams are added or concatenated on an element-by-element basis before the header is input. Compared to the FPN network used in YOLO v3, SPP can greatly increase the receptive field and separate the most significant context features at hardly any reduction in the network operating speed. In addition, YOLO v4 selects the path aggregation network (PANet) from different backbone layers as the parameter aggregation method for different levels of detectors. Therefore, YOLO v4 uses the modified versions of SPP, PAN, and self-attention-based deep learning method (SAM) to gradually replace FPN, retaining the rich spatial information from the bottom-up data stream and the rich semantic information from the top-down data stream.

In addition, YOLO v4 reasonably uses the bag of freebie and bag of special methods for tuning. Compared to YOLO v3, the average precision (AP) and FPS of YOLO v4 show an increase of 10% and 12%, respectively.

3.2. DeepSORT Vehicle Tracking Model. DeepSORT is an improved version of the SORT algorithm. It uses the KF prediction in the image space, uses the Hungarian algorithm to correlate the data frame-by-frame, and calculates the overlap rate of the bounding boxes from the correlation metric, which exhibits good performance at a high frame rate. Its specific process of dealing with tracking problems mainly includes trajectory processing and state estimation, information association, and cascade matching.

3.2.1. Trajectory Processing and State Estimation. An eight-dimensional space $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ is used for representing the state of a trajectory at a certain moment, where (u, v) represents the center coordinates of the predicted bounding box, h represents the height of the predicted target frame of a vehicle, and r refers to the aspect ratio of the image. The remaining four variables represent the speed information of each parameter relative to the image coordinates. A counter is set for each tracker of the target. If the tracking and detection results match each other, the tracker counter is reset to 0. If the tracker cannot find a matching result for a period of time, the tracker is deleted from the list. When a new detection result appears in a certain frame (that is, a detection result that cannot match the current tracking list appears),

a new tracker is created for the frame. If the prediction result of the new tracking target position matches the detection result for three consecutive frames, it is considered that a new target has appeared. Otherwise, it is considered that a "false alarm" has occurred, and the tracker is deleted from the tracker list.

3.2.2. Information Association. The Mahalanobis distance between the detection frame and the tracker prediction frame is used for describing the calculation of the degree of correlation of the target motion information:

$$l^{(1)}(i,j) = \left(l_j - p_i\right)^T X_i^{-1}\left(l_j - p\right), \tag{1}$$

where l_j represents the predicted position of the jth detection frame, p_j represents the predicted position of the target by the jth tracker, and X_i represents the covariance matrix between the detected position and the average tracking position. Taking into account the continuity of movement, the Mahalanobis distance matching method has been used in this work, and the 0.95 quantile of the χ^2 distribution has been used as the threshold. Considering that the Mahalanobis distance association method will be invalid when the camera is in motion, the target appearance information association has been introduced, and its process is as follows:

For each detection block, l_j, a feature vector, r_j, is calculated using a CNN network, and the condition $\|r_j\| = 1$ is imposed on it. A channel for each tracking target is constructed in order to store the feature vector of the last 100 frames successfully associated with each target. Following this, the minimum cosine distance between the latest 100 successfully associated feature sets of the ith tracker and the feature vector of the ith detection result of the current frame is calculated. If the distance is less than a certain threshold, the association is successful. The distance is calculated as follows:

$$l^{(2)}(i,j) = \min\left\{1 - r_j^T r_k \middle| r_k^{(i)} \in R_i\right\}. \tag{2}$$

The DeepSORT algorithm adopts the way of the fusion measurement and considers the information on the association of motion and object appearance at the same time. The two pieces of information are linearly weighted to calculate the degree of matching between the final detection and the tracking tracks using the following expression:

$$c_{i,j} = \lambda l^{(1)}(i,j) + (1-\lambda) l^{(2)}(i,j). \tag{3}$$

3.2.3. Cascade Matching. When a target is occluded for a long time, the uncertainty of filtering prediction will be greatly increased, and the observability of the state space will be greatly reduced. At this time, if two trackers compete for the matching right of the same detection result, the trajectory with a longer occlusion time is often blocked because the position information is not updated for a long time, thus increasing the uncertainty of the predicted position during tracking. In other words, the covariance will be larger, and the Mahalanobis distance will be smaller. Therefore, the detection result is more likely to be related to the trajectory having a longer occlusion time. This undesirable effect often destroys the continuity of tracking. The core idea of cascade matching is to match trajectories having the same disappearing time from small to large in order to ensure that the most recent target is given the greatest priority. Its specific process is shown in Figure 2.

The method developed in this work first uses YOLO v4 to detect the vehicle targets. Then, the tracking-by-detection DeepSORT algorithm is used to write the result of the frame of the traveling vehicle into the tracking queue for trajectory processing and state estimation. Finally, real-time tracking is done by information association and cascade matching. The overall flow of the algorithm is shown in Figure 3.

3.3. Vehicle Positioning Algorithm

3.3.1. Problem Description. During the detection and tracking of ground vehicles by the UAV, the vehicle is surrounded by a detection frame and marked with an ID. The center of the detection frame is taken as the target point, and its pixel coordinate is (x_p, y_p). The target line-of-sight vector, $\vec{\mathbf{r}}$, is defined as the vector between the optical center of the camera and the target point. As a result, $\vec{\mathbf{r}}$ can effectively reflect the relative position between the target point T and the UAV. The relationship between the parameters is shown in Figure 4.

The geographic coordinate system of the UAV is defined with the center of the GPS receiver as the origin, the X-axis pointing toward the true north direction, the Y-axis pointing toward the true east direction, and the Z-, X-, and Y-axis form a right-handed coordinate system. The line-of-sight angle is defined as (ρ, ε), where ρ is the angle between the target line-of-sight vector and the Z-axis and is called the line-of-sight height angle, ε is the angle between the projection of the target line-of-sight vector on the XOY plane and the X-axis, and is called the line-of-sight direction angle. During the flight of the UAV, the attitude angle of the UAV, the pointing of the camera, and the position of the target jointly determine the values of ρ and ε.

In order to calculate the target line-of-sight angle, three coordinate systems are defined, namely, the camera coordinate system (abbreviated as the C coordinate system, with the optical center of the camera as the origin), the inertial measurement unit (IMU), inertial platform coordinate system (abbreviated as the I coordinate system, with the IMU measurement center as the origin), and the UAV geographic coordinate system (abbreviated as the L coordinate system, with the center of the GPS receiver as the origin). The relationship between the spatial positions of the three coordinate systems is shown in Figure 5.

Let $\mathbf{t}_C = (x_p, y_p, f)^T$ be the coordinates of the target image point in the camera coordinate system, where f is the focal length of the camera. Assuming ϕ as the UAV yaw angle, γ as the pitch angle, θ as the roll angle, α as the azimuth angle of the camera, and β as the elevation angle, the coordinates of $\mathbf{t}_C$ in the geographic coordinates of the UAV can be obtained as

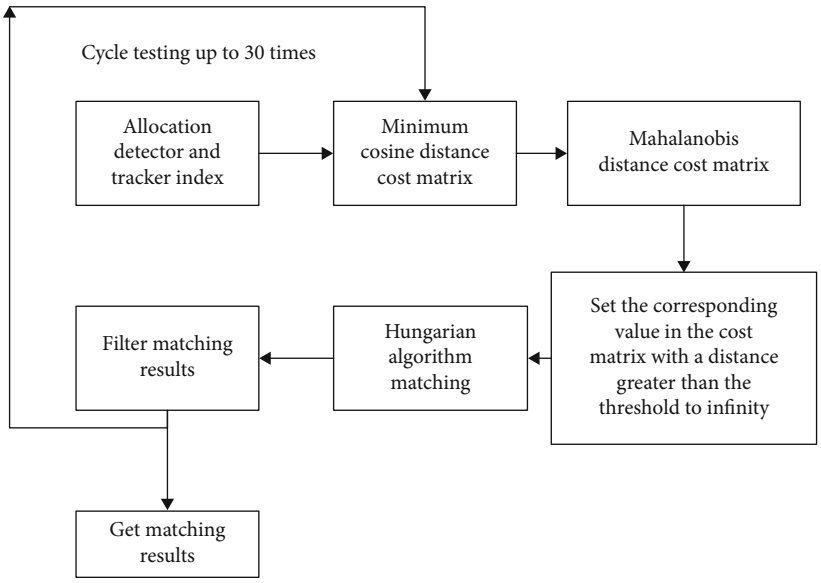


Figure 2: Schematic showing the cascade matching process.

$$\mathbf{t}_L = (x, y, z)^T = \mathbf{Rot}_I^L \cdot \mathbf{Rot}_C^I \cdot \mathbf{t}_C, \tag{4}$$

where $\mathbf{Rot}_C^I$ is the rotation matrix from the C series to the I series and $\mathbf{Rot}_C^I$ is determined by the azimuth angle, α, and the elevation angle, β, of the camera. $\mathbf{Rot}_I^L$ is the rotation matrix from the I series to the L series, and $\mathbf{Rot}_I^L$ is determined by the yaw angle,ϕ, of the UAV, the pitch angle,γ, and the roll angle,θ.

After obtaining $\mathbf{t}_L$ according to Equation (4), the line-of-sight angle (ρ,ε) can be calculated using the following equations:

$$\begin{aligned} \rho &= \tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{z}\right) \rho \in \left[0, \frac{\pi}{2}\right), \\ \varepsilon &= \tan^{-1}\left(\frac{y}{x}\right) \varepsilon \in (0, 2\pi]. \end{aligned} \tag{5}$$

In the process of discovering, tracking, and locating the target, the position coordinates of the UAV are (p_x, p_y, p_z), and the target coordinates are (t_x, t_y, t_z). Selecting the state variable $\mathbf{x}_k = [t_x, t_y, t_z]_k^T$ to represent the estimation of the target position, the discrete state equation of the system can be written as

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_{k+1,k}\mathbf{x}_k + \mathbf{w}_k, \tag{6}$$

where $\mathbf{\Phi}_{k+1,k}$ is the state transition matrix and $\mathbf{w}_k$ is the system noise matrix, $\mathbf{w}_k \sim N(0, Q_k)$.

The measurement equation of the system is

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \tag{7}$$

where $\mathbf{v}_k$ is the random noise in the measurement and its covariance matrix is R. The system noise, $\mathbf{w}_k$, and the measurement noise, $\mathbf{v}_k$, are uncorrelated zero-mean Gaussian white noise.

Using the triangular relationship between the UAV and the vehicle target point, we get

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \rho \\ \omega \\ r \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\dfrac{\sqrt{(p_x - t_x)^2 + \left(p_y - t_y\right)^2}}{p_z - t_z}\right) \\ \tan^{-1}\left(\dfrac{p_y - t_y}{p_x - t_x}\right) \\ \sqrt{(p_x - t_x)^2 + \left(p_y - t_y\right)^2 + (p_z - t_z)^2} \end{bmatrix}, \tag{8}$$

where r is the distance between the UAV and the target. In an urban environment, due to the flat terrain, this value can be calculated from the relative height of the UAV to the ground and ρ.

3.3.2. IMM-PF Algorithm. In actual practice, the state of the vehicles changes dynamically, and this is difficult to describe with a single motion model. On the one hand, the UAV target positioning task is mainly composed of three major systems: the aircraft, the camera, and the global positioning system (GPS)/inertial navigation system (INS). The GPS measurement has errors in estimating the latitude and longitude of the aircraft, and the INS also has errors in the measurement of the attitude of the aircraft. In addition, the sight axis of the camera also has a jitter. For moving targets, the PF algorithm can be used, which can handle nonlinear and non-Gaussian system filtering. In this work, the advantages of the IMM and the PF algorithm have been combined, and the IMM-PF algorithm has been employed to achieve target positioning.

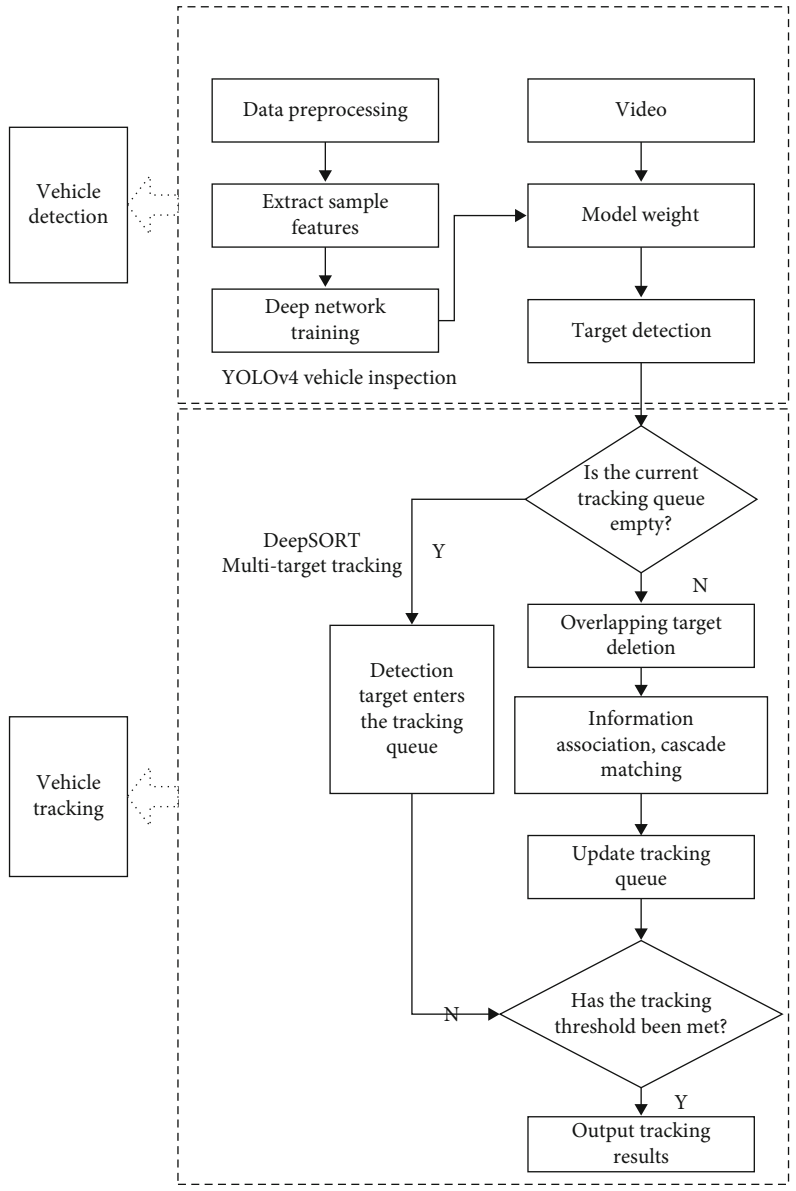


Figure 3: Flowchart of the vehicle detection and tracking.

For multiple models, the state transition equation and observation equation are

$$x_k = F(m_k)x_{k-1} + G(m_k)u_{k-1}(m_k), \tag{9}$$

$$z_k = H(x_k, m_k) + v_k(m_k). \tag{10}$$

In the above equations, x_k represents the target state vector of the model m_k at time k, and z_k represents the corresponding state observation variable. The state transition matrix, F, the observation matrix, H, the process noise, u_k, and the observation noise, v_k, are all related to the model m_k. The probability density of u_k and v_k is defined as $d_{u_k(m_k)}(u)$ and $d_{v_k(m_k)}(v)$, respectively.

Assuming that there are a limited number of system models, $m_k \in M$, and the model probability is $u_k(m_k)$, the transition probability between the models can be represented by a Markov chain as follows:

$$\Pr\{m_k = j | m_{k-1} = i\} = p_{ij}, \forall i, j \in M, \tag{11}$$

where p_{ij} represents the probability that the model m_{k-1} at the time $k-1$ transfers to the model m_k at time k, assuming that it remains unchanged during the tracking process.

Assuming that the initial value of the state x_0 is known, the initial model probability, $\{\mu_0(m_k)\}_{m_k=1}^{M}$, and the observed value, $z_{1:k}$, at each time are known, the posterior probability density, $\widehat{p}(x_k|z_{1:k})$, of the state at that time is estimated, and subsequently, the estimated value, $\widehat{x}_k$, of the system state is obtained.

Using the IMM algorithm as the basic framework, the IMM-PF algorithm uses PF as the model matching filter. The IMM algorithm is divided into four steps: input

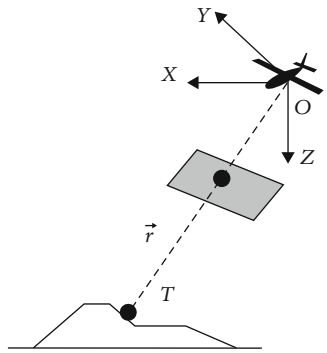


Figure 4: Schematic showing the line-of-sight angle.

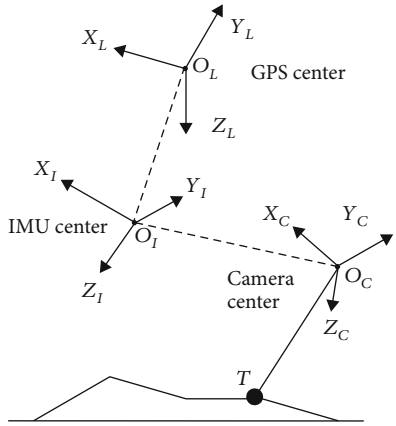


Figure 5: Coordinate conversion.

interaction, model matching filtering, model probability update, and estimated output. Taking the IMM algorithm as the basic framework, recursive Bayesian filtering is used for deriving the IMM-PF algorithm from time $k-1$ to time k.

(1) Input Interaction. First, the interaction probability of the system model at time $k-1$ is calculated using the following expression:

$$\mu_{k-1}(m_{k-1}|m_k) = \frac{p_{ij}\mu_{k-1}(m_{k-1})}{b_{k-1}(m_k)}. \tag{12}$$

The normalization factor is given by

$$b_{k-1}(m_k) = \sum_{m_{k-1}\in M} p_{ij}\mu_{k-1}(m_{k-1}). \tag{13}$$

The particles in each model interact with the state estimates of the other models ($l = 1, 2, \cdots, N$):

$$\tilde{x}_{k-1}^{l}(m_k) = \sum_{m_{k-1}\neq m_k}^{M} \tilde{x}_{k-1}(m_{k-1})\mu_{k-1}(m_{k-1}|m_k) + \tilde{x}_{k-1}^{l}(m_k)\mu_{k-1}(m_k|m_k). \tag{14}$$

(2) Interactive Model Matched Filtering. The particle state at time k is predicted by the state transition equation (Equation (9)) as

$$\tilde{x}_k^l(m_k) = F(m_k)\tilde{x}_{k-1}^l(m_k) + G(m_k)\tilde{\mu}_{k-1}^l(m_k). \tag{15}$$

The observed value of the particle state at time k is predicted by the observation equation (Equation (10)) as

$$\tilde{z}_k^l(m_k) = H\left(\tilde{x}_k^l, m_k\right). \tag{16}$$

The particle weight is obtained from the system state observation value, z_k, and the observation noise probability density, $d_{v_k(m_k)}(v)$, as

$$\tilde{w}_k^l(m_k) = d_{v_k(m_k)}\left(z_k - \tilde{z}_k^l(m_k)\right). \tag{17}$$

Weight normalization is expressed as

$$\tilde{w}_k^l(m_k) = \frac{\tilde{w}_k^l(m_k)}{\sum_{l=1}^{N}\tilde{w}_k^l(m_k)}. \tag{18}$$

$\tilde{x}_k^l(m_k)$is resampled using the expression Pr $[\bar{x}_k^l(m_k) = \tilde{x}_k^l(m_k)] = \tilde{w}_k^l(m_k)$ to obtain a new particle set, $\bar{x}_k^l(m_k)$ and set the particle weight is $\bar{x}_k^l(m_k) = 1/N$. Then, the estimated state of the model m_k at time k is

$$\tilde{x}_k^l(m_k) = \frac{1}{N}\sum_{l=1}^{N}\bar{x}_k^l(m_k). \tag{19}$$

(3) Model Probability Update. The residual error of particle observations is calculated by

$$r_k^l(m_k) = z_k - H\left(\bar{x}_k^l, m_k\right). \tag{20}$$

The mean of the particle observations is calculated by

$$\bar{z}_k(m_k) = \frac{1}{N}\sum_{l=1}^{N}H\left(\bar{x}_k^l, m_k\right). \tag{21}$$

The residual covariance is

$$S_k(m_k) = \frac{1}{N}\sum_{l=1}^{N}\left[H\left(\bar{x}_k^l, m_k\right) - \bar{z}_k(m_k)\right]\cdot\left[H\left(\bar{x}_k^l, m_k\right) - \bar{z}_k(m_k)\right]^T. \tag{22}$$

The likelihood function of the model is expressed by

$$\Lambda_k(m_k) = \frac{1}{N}\sum_{l=1}^{N}N\left(r_k^l(m_k); 0, S_k(m_k)\right). \tag{23}$$

Model probability is updated using

$$\mu_k(m_k) = \frac{\Lambda_k(m_k)b_{k-1}(m_k)}{B_k}, \tag{24}$$

where

$$B_k = \sum_{m_k \in M} \Lambda_k(m_k) b_{k-1}(m_k). \tag{25}$$

(4) Estimated Output. The estimated state of the target is calculated using the following expression:

$$\hat{x}_k = \sum_{m_k \in M} \hat{x}_k(m_k) \mu_k(m_k). \tag{26}$$

The interaction, filtering, estimation, and resampling of the IMM-PF algorithm are based on particles.

4. Simulation Tests and Analysis

4.1. Object Detection and Tracking Tests. The dataset used in this study consists of aerial images of road traffic in an urban environment selected from the VisUAV multiobject aerial photography dataset. Most of the labeled vehicle objects in the dataset are cars, buses, trucks, and vans, with a total of 15741 images, which are used as the training dataset for the YOLO v4 network. Subsequently, VisUAV2019-MOT is used as the benchmark dataset to test the algorithm of this study. VisUAV2019-MOT is a video sequence acquired by an unmanned aerial vehicle (UAV), covering different shooting perspectives as well as different weather and lighting conditions. On average, each frame contains multiple-detection frames, while each sequence contains multiple objects. The resolution of the video sequence is 2720×1530.

In this work, the model is trained and tested on a platform with Intel Core i7-8700 k CPU@3.7GHz, 32GB RAM, and GeForce GTX 2080 8GB GPU. The operating system is Ubuntu 16.04. The supporting environments are Python 3.5.2, Keras = 2.1.3, TensorFlow − gpu = 1.4.0, and OpenCV − python = 3.4.3.

Figure 6 presents sample images of scenes from the VisUAV2019-MOT dataset. The VisUAV2019-MOT dataset contains several complex scenes, such as highway entrances, pedestrian streets, roads, and T-junctions. The scenes have high-traffic flow and a large number of vehicle objects with changing motion characteristics. Furthermore, the moving UAV shots can fully reflect whether the performance of the algorithm is satisfactory.

In this work, the following evaluation criteria are used to analyze the advantages and disadvantages of multiobject tracking algorithms for different cases.

Multiple object tracking accuracy (MOTA) is an intuitive measure of tracking the performance of detecting objects and maintaining trajectories, independent of the estimated accuracy of the object position. The larger its value, the better the performance. It is calculated as follows:

$$\mathrm{MOTA} = 1 - \frac{\sum_t \mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDSW}_t}{\sum_t \mathrm{GT}_t}, \tag{27}$$

where FN_t is false negative, FP_t is false positive, IDSW_t is ID Switch, and GT_t is the number of all objects.

Multiple object tracking precision (MOTP) indicates the positioning accuracy. The larger the value, the better. It is calculated as follows:

$$\mathrm{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t C_t}, \tag{28}$$

where d is the average metric distance (i.e., the IoU value of the bounding box) and C denotes the number of successful matches for the current frame.

Mostly tracked (MT) denotes the number of successful tracking results that match the true value at least 80% of the time.

Mostly lost (ML) represents the number of successful tracking results that match the true value in less than 20% of the time.

ID switch indicates the number of times the assigned ID has jumped.

FM (fragmentation) indicates the number of times in which the tracking was interrupted (i.e., the number of times the tagged object was not matched).

FP (false positive) is the number of false alarms, referring to the trajectory of false predictions.

FN (false negative) is the number of missed detections and undetected tracking objects.

Based on the above eight metrics, the trained YOLO v4 vehicle detection model is used as a detector in this section. Further, video sequences with different viewing angles and lighting are selected from the VisUAV2019-MOT dataset to test the tracking algorithm. The evaluation results are presented in Table 1.

From Table 1, the MOTP values of the algorithms in this work are relatively high, and all remain above 78%. The positioning accuracy of video sequence uav0000306_00230_v reaches 84.4%, which further demonstrates the satisfactory performance of the detector trained. In addition, the lowest ID jump value in video sequence uav0000077_00720_v is only 46, and the test values of false alarm number and missed detection number vary widely among video sequences, because of different sequences shooting backgrounds and number of vehicles.

Figure 7 shows the tracking results for a video sequence of a road intersection. From the figure, the proposed algorithm shows satisfactory results in a complex environment. The algorithm not only accurately achieves the detection of multiple vehicle models for multiple objects in each frame but also establishes the correspondence with the object when performing tracking for the same vehicle.

4.2. Simulation of the Object Positioning Algorithm. In this work, a maneuvering object is simulated for positioning to verify the feasibility and effectiveness of the algorithm. The object alternately performs constant velocity motion, constant turn motion, and constant acceleration motion. The system process noise u_k and observation noise v_k are both Gaussian white noises. The sampling period is $T = 0.1$ s, and the simulation time is 40 s. Figure 8 shows the trajectory of the maneuvering object.

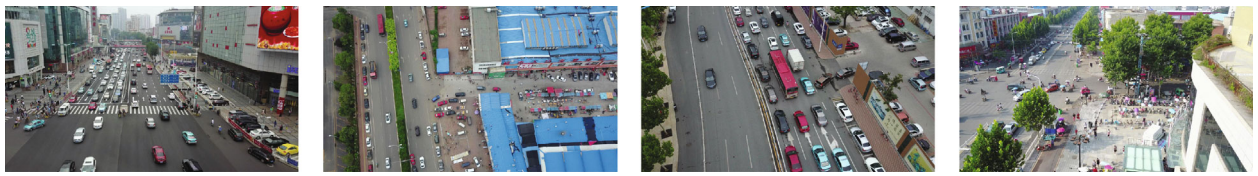
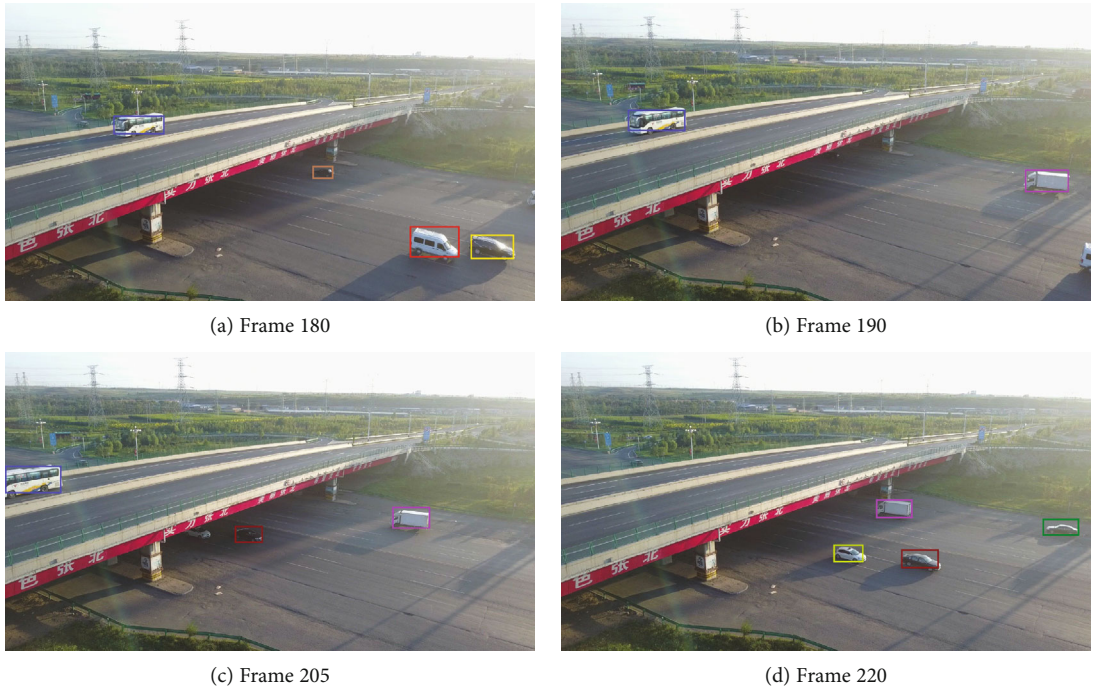


Figure 6: Selected scenes from the VisUAV2019-MOT dataset.

Table 1: Evaluation results of different types of video sequence tracking.

Video sequences	MOTA	MOTP	MT	ML	ID switch	FM	FP	FN
uav0000077_00720_v	75.1	82.2	20	20	46	94	241	122
uav0000119_02301_v	74.2	80.5	25	18	61	102	326	301
uav0000188_00000_v	65.3	78.3	41	16	134	122	177	266
uav0000249_02688_v	72.2	83.2	21	19	77	81	220	141
uav0000306_00230_v	76.8	84.4	34	17	131	159	196	212



(a) Frame 180

(b) Frame 190

(c) Frame 205

(d) Frame 220

Figure 7: Tracking results of the proposed algorithm.

The CV-EKF and IMM-EKF algorithms are used for comparison with the IMM-PF algorithm used in this paper. A total of 50 Monte-Carlo simulation experiments are performed, where the number of particles used in the PF algorithm is $N = 100$.

The positon estimates of the three algorithms are shown in Figure 9, and a comparison of the RMSE curves of locations estimated by the three algorithms is shown in Figure 10. The figure shows that IMM-PF has significantly better tracking accuracy for maneuvering objects than CV and IMM-EKF algorithms. The EKF based on the single CV model can hardly track the object effectively when the maneuvering object turns, resulting in a larger distance error; the IMM-EKF can track the object when the object turns due to the IMM, but the distance error is larger than that of the IMM-PF. The IMM-PF filtering algorithm can handle nonlinear motions better, which makes the algorithm maintain stable tracking of the object under nonlinear motions. The simulation experiments show that the IMM-PF filtering algorithm has a smaller RMSE than the IMM-EKF algorithm and thus has better performance for nonlinear positioning.

To represent the effect of the IMM filter, the model probabilities are plotted as a function of time, as shown in Figure 11. The figure shows three initialized models with the same probability, which quickly converge to the CV model as the filter is updated. After 40 s of motion, the CV model no longer holds true, and the probability of the CT model becomes very high. In the final time of motion, the

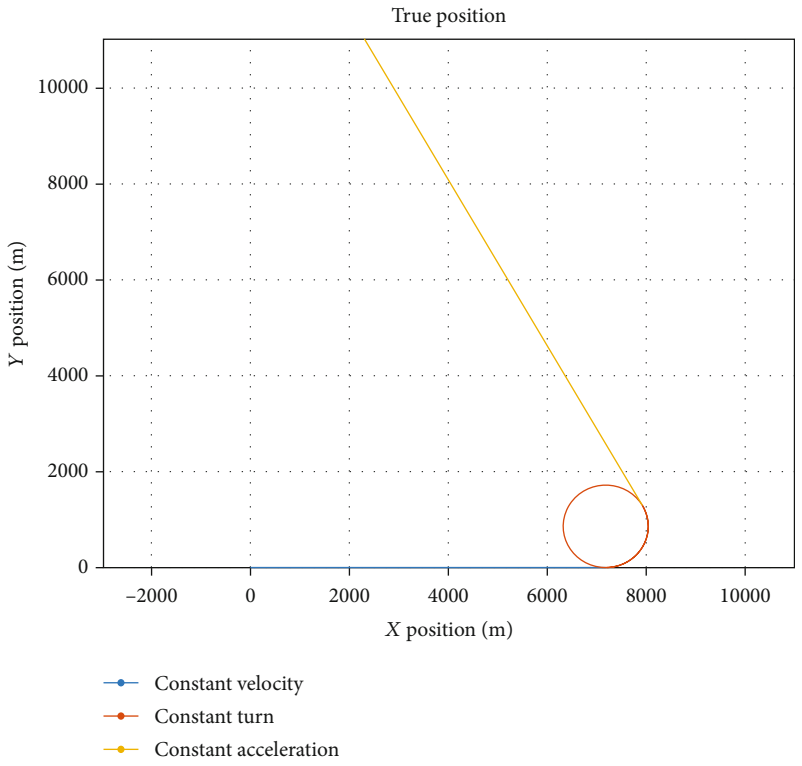


Figure 8: Trajectory of the maneuvering object.

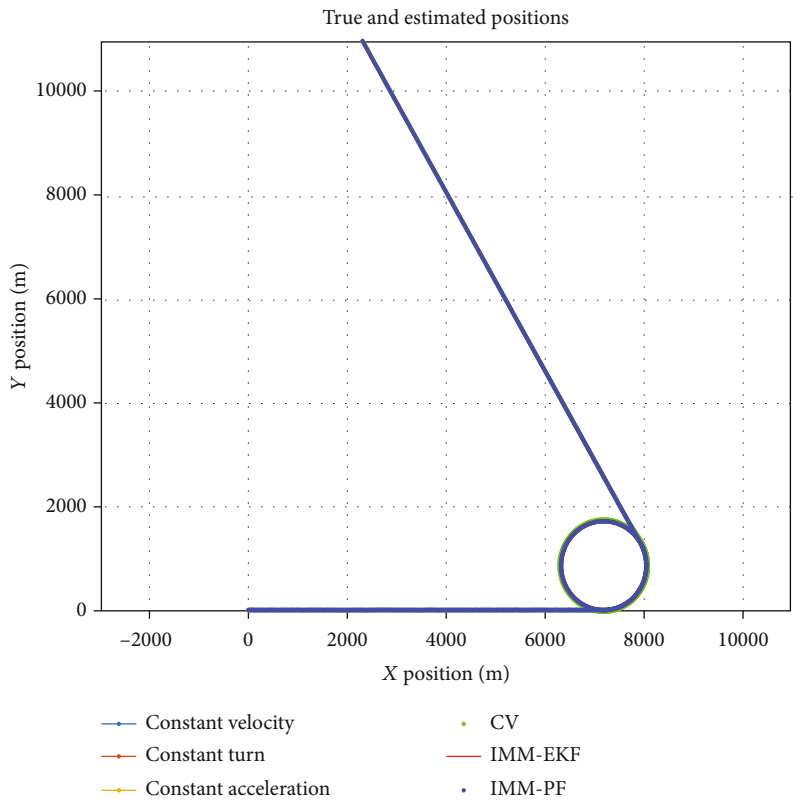


Figure 9: Position estimation.

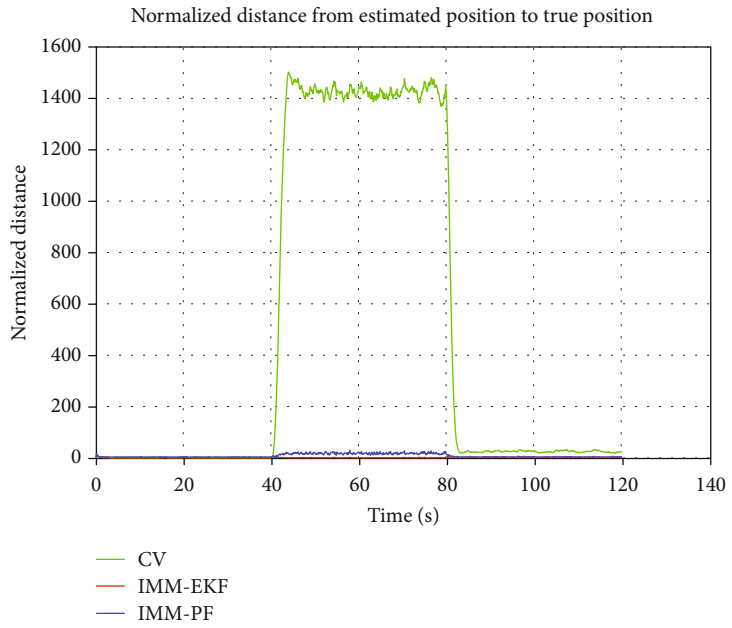


Figure 10: Comparison of RMSEs of positions estimated by the three algorithms.

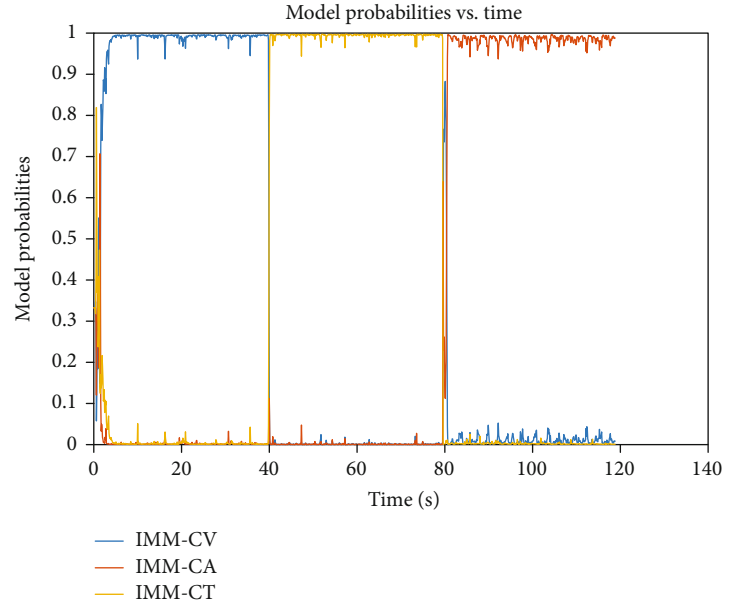


Figure 11: Motion model probability switching.

CA model obtains the highest probability. The switching of multiple motion models verifies the effectiveness of the IMM filter.

5. Conclusion

This paper investigated related technology to address the need for automatic UAV-based vehicle detection, tracking, and positioning. The YOLO v4 object detection algorithm was used as the basis to train a vehicle detection network from the UAV perspective. At the object tracking stage, the DeepSORT algorithm was adopted. The combination of YOLO v4 and DeepSORT algorithms effectively improves the accuracy and robustness of multivehicle detection and tracking in complex urban scenes. The particle filtering and IMM algorithms were combined and applied to the UAV for positioning of maneuvering objects, which can improve the target positioning accuracy effectively.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer, "UAV-enabled intelligent transportation systems for the smart city: applications and challenges," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 22–28, 2017.

[2] H. Kim, L. Mokdad, and J. Ben-Othman, "Designing UAV surveillance frameworks for smart city and extensive ocean with differential perspectives," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 98–104, 2018.

[3] X. Liu and X. Zhang, "NOMA-based resource allocation for cluster-based cognitive industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5379–5388, 2020.

[4] X. Liu, X. Zhai, W. Lu, and C. Wu, "QoS-guarantee resource allocation for multibeam satellite industrial internet of things with NOMA," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2052–2061, 2021.

[5] X. Liu and X. Zhang, "Rate and energy efficiency improvements for 5G-based IoT with simultaneous transfer," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 5971–5980, 2019.

[6] M. Quigley, M. A. Goodrich, S. Griffiths, A. Eldredge, and R. W. Beard, "Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2600–2605, Barcelona, Spain, 2005.

[7] D. Decarlo and D. Metaxas, "Optical flow constraints on deformable models with applications to face tracking," *International Journal of Computer Vision*, vol. 38, no. 2, pp. 99–127, 2000.

[8] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: multitarget detection and tracking," in *Lecture Notes in Computer Science*, pp. 28–39, Springer, Berlin, Heidelberg, 2004.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, pp. 142–149, Hilton Head, SC, USA, 2000.

[10] D. Exner, E. Bruns, D. Kurz, A. Grundhöfer, and O. Bimber, "Fast and robust CAMShift tracking," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pp. 9–16, San Francisco, CA, USA, 2010.

[11] H. Moridvaisi, F. Razzazi, M. A. Pourmina, and M. Dousti, "An extended KCF tracking algorithm based on TLD structure in low frame rate videos," *Multimedia Tools and Applications*, vol. 79, no. 29-30, pp. 20995–21012, 2020.

[12] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[13] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: a survey," *Neurocomputing*, vol. 300, pp. 17–33, 2018.

[14] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, "Learning by tracking: Siamese CNN for robust target association," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 33–40, Las Vegas, NV, USA, 2016.

[15] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4836–4845, Venice, Italy, 2017.

[16] L. Jiang, M. Cheng, and T. Matsumoto, "A TOA-DOA hybrid factor graph-based technique for multi-target geolocation and tracking," *IEEE Access*, vol. 9, pp. 14203–14215, 2021.

[17] W. Li, S. Sun, Y. Jia, and J. Du, "Robust unscented Kalman filter with adaptation of process and measurement noise covariances," *Digital Signal Processing*, vol. 48, pp. 93–103, 2016.

[18] J. Gao and H. Zhao, "An improved particle filter for UAV passive tracking based on RSS," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pp. 1–6, Victoria, BC, Canada, 2020.

[19] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 1515–1522, Kyoto, Japan, 2009.

[20] D. Magill, "Optimal adaptive estimation of sampled stochastic processes," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 434–439, 1965.

[21] H. A. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[22] Y. Boers and J. N. Driessen, "Interacting multiple model particle filter," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 150, no. 5, pp. 344–349, 2003.

[23] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: optimal speed and accuracy of object detection," 2020, https://arxiv.org/abs/2004.10934.

[24] B. Veeramani, J. W. Raymond, and P. Chanda, "DeepSort: deep convolutional networks for sorting haploid maize seeds," *BMC Bioinformatics*, vol. 19, no. 9, pp. 1–9, 2018.