Received June 6, 2021, accepted June 17, 2021, date of publication June 23, 2021, date of current version June 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091871

Deformable Model-Based Vehicle Tracking and Recognition Using 3-D Constrained Multiple-Kernels and Kalman Filter

TAO LIU AND YONG LIU
Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Post and Telecommunications, Beijing 100876, China
Beijing Laboratory of Advanced Information Networks, Beijing University of Post and Telecommunications, Beijing 100876, China
School of Information and Communication Engineering, Beijing University of Post and Telecommunications, Beijing 100876, China

Corresponding author: Tao Liu (liutao19890403@163.com)

This work was supported in part by the 111 Project under Grant B17007, and in part by the China Scholarship Council Funding.

ABSTRACT Video-based vehicle tracking and recognition is an important application in the Intelligent Transportation System. High similarity among vehicle types, frequent occlusion and low video resolution in traffic surveillance are the major problems in this research area. In this paper, we proposed a vehicle tracking system by using 3-D constrained multiple-kernels, facilitated with Kalman filtering, to continuously update the location of the moving vehicles. To further robustly and efficiently track vehicles that are partially or even fully occluded, evolutionary optimization is applied to camera calibration for systematically building 3-D vehicle model, from which we can extract the vehicle's features such as the vehicle type, color and license plate. Then, a self-similarity descriptor is further introduced for vehicle re-identification. The proposed system is evaluated on the NVIDIA AI City Datasets and one self-recorded high-resolution video. The experimental results have shown the favorable performance, which not only can successfully track vehicles under occlusion, but also can maintain the knowledge of 3-D vehicle geometry.

INDEX TERMS 3-D vehicle modeling, constrained multiple kernels, camera calibration, Kalman filter, vehicle tracking and recogniton.

I. INTRODUCTION

Currently, video-based traffic surveillance of vehicles plays an important role in intelligent transportation systems(ITS). In traffic surveillance, vehicle tracking is a key technology that can be applied in many applications. By tracking a vehicle, its trajectory can be collected in the video for advanced analysis, such as predicting and avoiding accidents, detecting stolen vehicles, calculating the traffic flow, and collecting traffic statistics for further decision-making. Therefore, researchers are motivated to develop a robust vehicle tracking system, which can not only track vehicles effectively, but also be able to collect the information for use in traffic management.

Vehicle tracking and recognition in urban environments is challenging due to several reasons. First, due to the color information is not very discriminative between the different types of vehicle, the appearance similarity among vehicles is generally high. Second, frequent occlusion by other vehicles, trees and lighting posts at traffic intersections will cause severe identity switches. Last but not least, the low video resolution and high distortion nature caused by the dynamic viewpoints of cameras in traffic surveillance leads to clear license plate images for recognition is hardly available in general cases. Many approaches based on color histogram or 2-D contours have thus been proposed for vehicle tracking [1]–[4]. On the other hand, modularizing a vehicle into a 3-D vehicle model is an intuitive way to extract features for vehicle identification. Several methods which use a 3-D vehicle model to track and recognize vehicle have also been proposed [5]–[9]. These model-based methods show their efficient performance in vehicle tracking and recognition without explicitly handling occlusion scenarios.

In this paper, we effectively combine the constrained multiple kernel(CMK) tracking technique in 3-D space with the 3-D vehicle model into one system, which is based on the Kalman filter framework. With the help of the 3DCMK, the proposed tracking system can successfully track the

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Gao.

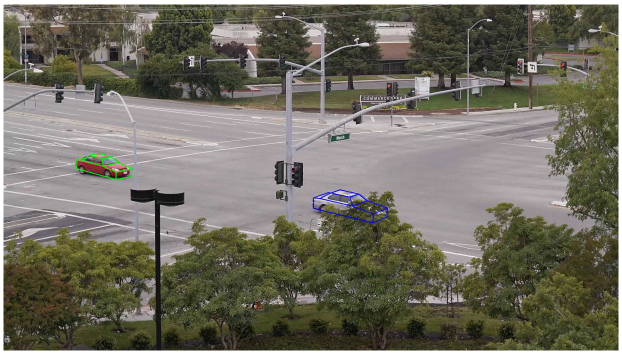


FIGURE 1. The result of multiple vehicles tracking under partial occlution with the 3-D vehicle model.

vehicle under occlusion. On the other hand, after building a 3-D vehicle model for each tracked vehicles, the system can easily extract the features of the vehicle, such as vehicle's type, color and license plate, and then a self-similarity descriptor (SSD) [10] is adopted to measure the similarity between license plate images for vehicle re-identification. Facilitated by the 3-D vehicle model, the proposed system can robustly handle the partial occlusion by model-based CMK tracking, and total occlusion by performing license plate matching with the help of SSD. Figure 1 shows the result of multiple vehicles tracking under partial occlusion with the 3-D vehicle model.

The remainers of this paper is organized as follows: Section II gives a brief survey on the related work. In Section III, we describe the proposed tracking framework in detail. Specifically, we discuss vehicle detection and classification, evolutionary optimization for camera calibration, the fitness procedure of 3-D vehicle modeling and vehicle features matching. Section IV depicts how to integrate the 3-D vehicle model and the 3DCMK tracking into the Kalman filter framework. The experimental results are demonstrated in Section V. followed by the conclusion in Section VI.

II. RELATED WORK

Vehicle tracking is a crucial and unavoidable task in the field of intelligent transportation systems including video surveillance, traffic control and unmanned vehicle control. Many approaches focus on vehicle tracking have thus been proposed by the researchers.

In general, the existing vehicle tracking approaches [11] using surveillance camera can be roughly divided into two categories: 2-D based and 3-D based approaches. In 2-D based approaches, the 2-D motions and trajectories of the vehicle are commonly used in the particle-filtering or Kalman-filtering frameworks, which mainly based on the 2-D geometry features, such as contours, lines, and edges [12]–[14]. The approach in [15] combined an edge gradient-based shape feature and color histogram appearance feature to track under a sequential Monte Carlo framework. The method in [16] use size, linearity features and a set of rules to roughly obtain the vehicle silhouette. In [2] a 2-D based vehicle contour tracking method was proposed under the well-known particle filter condensation algorithm. The approach in [4] proposed an adaptive particle-filtering (APF) algorithm for dealing with partial and total occlusions scenarios. Leibe *et al.* [17] propose a method to extract the local features around the interest points, which are matched with those features from a previously learned codebook. Several literatures adopt corners or interest points as tracking feature, such as SIFT [18] and HOGs [11], for these feature descriptor are sensitive to intensity, lines and corners. Alternatively, some approaches [19], [20] introduce color histogram as tracking features, since it is not only more invariant to vehicle translations and rotations but also more robust to noise and occlusions.

The 2-D based vehicle tracking can be regarded as a specific category of object tracking in the surveillance video, which has been extensively discussed and developed. Comaniciu *et al.* [21] proposed a framework called kernel-based object tracking has received great attention in the past few years due to the low computation and the spatially weighted color histogram-based features. The main concept for the framework is to maximize the similarity between the candidate's and the target's appearance model (i.e., color histogram) by building the spatially mask for the object with a kernel function. With this framework, many methods have been proposed to improve tracking performance [22]–[24]. Furthermore, the multiple kernel tracking technique, extended from the single kernel approaches by using certain predefined constraints, has also been proposed [25], [26]. Chu *et al.* [27] generalize the constrained multiple-kernel (CMK) tracking by adaptively adjusting kernels' weights, in order to efficiently compensate for the adverse effect caused by the occlusion.

Nevertheless, vehicle tracking is still challenging due to the fact that texture or color information is not very discriminative among different type of vehicles, so that these object-tracking methods may not be directly applied for vehicle tracking. In addition, it is not sufficient to track vehicles accurately by only using the color or texture information. For this reason, researchers have proposed a method of modularizing vehicle into 3-D model for vehicle tracking. Haag and Nagel *et al.* [28] combined the edge and corresponding features to build 3-D model, and then an extended Kalman-filtering framework was used for tracking. Ferryman *et al.* [29] develop a deformable model with 29 parameters and a principal component analysis framework is used for vehicle tracking and recognition. Tan and Baker [30] use the image gradient to estimate the orientation of a vehicle and the intensity values to evaluate the vehicle poses. Zhang *et al.* [7], [31] present a model based vehicle tracking system which adopt local gradient-based method to evaluate the fitness between the image data and the vehicle model projection, a novel evolutionary computing framework is also combined to estimate the 12 shape and three pose parameters iteratively. The approach in [32] propose an efficient search method using 3×3 search kernel

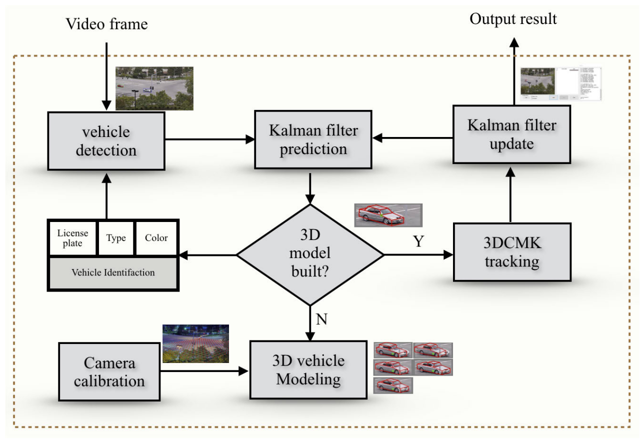


FIGURE 2. **Overview of the proposed system.**

and accelerated step length, to reduce the computational cost for the tracked poses.

Although the above approaches can successfully track the vehicle and achieve good experimental results in limited scenarios, they are specifically focus on the vehicle shape fitting, without considering the color information to perform tracking. In addition, the multiple kernels tracking is based on the 2-D (image) space, while the 3D vehicle model is defined in the 3-D space, the constraints for the multiple kernels in 3-D space will change during the movement of a vehicle owing to the varying view aspects. It may gradually make the tracking features of the vehicle unreliable, thereby making the error of tracking result larger and larger. To overcome this issue, we proposed a 3-D constrained multiple-kernels based vehicle tracking system which also adopt the 3D vehicle model technique. The proposed approach tracks the vehicle in 3-D space directly rather than track kernels in 2-D space. Owing to the 3-D constraints, the multiple kernels of vehicle are conditionally bound and mean-shifted properly according to the color information back projected from the image data. Meanwhile, a gradient-based method is developed to accurately fit the shape of the projected 3-D vehicle model. Therefore, the proposed approach not only builds the 3-D vehicle models according to the shape fitness evaluation, but also successfully tracks the vehicle while maintaining the knowledge of 3-D geometry.

III. OVERVIEW OF THE PROPOSED SYSTEM

Figure 2 shows the overview of the proposed system for vehicle tracking based on 3-D constrained multiple kernels(3DCMK) tracking and 3-D deformable vehicle modeling. The proposed system begins with vehicle detection, the state-of-the-art pre-trained YOLOv3 detector [33] is introduced to detect the vehicle and classify other objects like bicycle, motorcycle and pedestrian in the intersections. After that, the Kalman filter prediction is performed for each detected vehicle in the previous frames. The next stage is to detect whether the vehicle has been built a 3-D model or not. If it is not, the proposed system automatically builds a 3-D vehicle model for each detected vehicle. Meanwhile, evolutionary optimization is applied to static camera calibration for systematically and reliably building the detected vehicle a 3-D vehicle model, from which we can extract the vehicle features like the vehicle's type, color and license plate. On the other hand, a self-similarity descriptor is further introduced to measure the similarity of the license plate between the adjacent frames. Then these vehicle features will be stored as a feedback to identify a newly occurring vehicle in the surveillance video. At last, by using the measurement from the constrained multiple-kernel tracking results, we are able to update the Kalman filter and get the reliable vehicle tracking results. Each module is briefly described in the following subsection.

A. VEHICLE DETECTION AND CLASSIFICATION

Usually, Vehicle detection under a static camera basically follows these two steps: foreground segmentation and vehicle classification. The task of foreground segmentation is to extract the regions of interest from the image frame and avoid extracting as many background regions as possible. Then, vehicle classification is used to classify the extracted blobs as vehicle or non-vehicle.

The method of foreground segmentation under a static camera has been well developed. Such as the background difference method which mainly uses the color, edge, intensity, and gradient orientation of the target image and the background image to extract the foreground information. The inter-frame difference method and the optical flow method utilize the motion between adjacent frames to minimize the inclusion of the background region and extract foreground blobs by considering the size, position and aspect ratio. The method of the vehicle classification can be roughly divided into two categories: the template-based and the appearance-based method. The basic concept of the template-based vehicle classification is to perform a comparison between the predefined vehicle template and the same size regions of the image frame. The deformable part model(DPM) [34] is a classical template-based method, which using a root and several part templates to describe different partitions of a vehicle. The part templates are spatially connected with the root template according to the predefined geometry, so as to precisely depict vehicle. The appearance-based methods first extract the image features by using a pre-defined feature detectors such as Scale-invariant feature transform(SIFT), histogram of oriented gradients(HOG), local binary patter(LBP). Then, importing the extracted feature to a vehicle classifier, which is pre-trained by positive samples(vehicle) and negative samples(non-vehicle) with various learning algorithms, such as support-vector machine(SVM), AdaBoost, Convolutional Neural Network(CNN) and etc. The classifier will classify the extracted object a vehicle or non-vehicle.

In this paper, we mainly focus on the vehicle 3-D modeling and tracking. In order to avoid detecting other false moving objects at the intersection, such as pedestrians, bicycles, motorcycles, and etc. We adopt the state-of-the-art

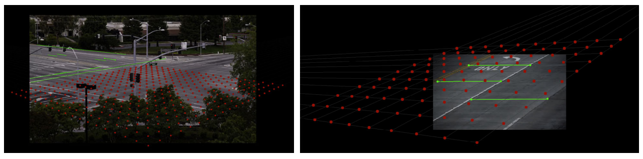


FIGURE 3. Visualization of the performance of the evolutionary camera calibration.

pre-trained YOLOv3 detector, which using the most advanced convolutional neural network technology to train the classifier, to help detecting vehicles. The vehicle detector can be embedded independently in the proposed system, so as to functionally perform vehicle detection.

B. EVOLUTIONARY OPTIMIZATION FOR CAMERA CALIBRATION

Camera calibration plays an important role in the many computer vision applications, such as 3-D objects tracking and localization. It is applied to calculate the intrinisic and extrinsic parameters of the camera, thus the projection matrix from the 3-D points in the model coordinate system (MCS) to the 2-D points in the image coordinate system (ICS) can be constructed. In each camera view, we manually label two pairs of parallel line on the 3D ground plane that are orthogonal with each other, from which we can obtain two vanishing points V_x and V_y on the ground plane. As proved in [35], the camera parameters in a 3×4 projection matrix P can be estimated from the two vanish points V_x and V_y with some assumptions on the intrinsic parameters of camera. To relax these assumptions for more accurate estimation of the camera parameters, we formulate the problem of optimization by minimizing the reprojection error. A set of line segments are manually selected on the ground plane, each is defined by two 3-D endpoints, noted P_k and Q_k, and the 3D ground truth lengths are pre-measured. The 2d endpoints of the line segments, noted p_k and q_k, can be backprojected to 3D points in MCS by using the calculated camera parameters. Euclidean distance is used to represent the estimated 3D lengths of the line segment, and the absolute differences between the ground truths and the estimations 3D length of the line segment are summed up to describe the reprojection error. Thus, the task of the optimization problem is defined as follows.

$$\min_{P} \sum_{k=1}^{N} \left| \|P_k - Q_k\|_2 - \left\| \hat{P}_k - \hat{Q}_k \right\|_2 \right|$$
$$\text{s.t.P} \in \text{Ran}_p, \quad p_k = P \cdot \hat{P}_k, q_k = P \cdot \hat{Q}_k \tag{1}$$

where $\hat{P}_k$ and $\hat{Q}_k$ represent the endpoints of the selected line segments backprojected to the 3D ground plane. Ran_P is the initial range of the camera parameters to be optimized.

Thanks to the classic evolutionary optimization algorithm, named Estimation of Distribution Algorithm (EDA) [36], the non-linear optimization problem in (1) can be iteratively solved to optimize the 11 camera parameters in the projection matrix P. Figure 3 shows the visualization of the camera calibration performance on a scene of the *NVIDIA AI City Dataset*(a) and the self-recorded video dataset(b). The uniformly distributed projected virtual grids on the ground plane are plotted as red dots, and the green solid lines present the manually selected line segments used to measure the re-projection error on the ground plane.

TABLE 1. Shape parameters of generic vehicle model (in millimeter).

Parameters	Descriptions	Value Range
L	Distance from 0 to 1	[3200,5500]
$W1$	Distance from 1 to 2	[1500,2100]
$W2$	Distance from 13 to 14	[1000,W1]
$H1$	Distance from 1 to 5	[400,800]
$H2$	Distance from 0 to 4	[400,800]
$H3$	Distance from 8 to I	[max(H1,H2),900]
$H4$	Distance from 13 to I	[$1.5 \times H3$, 1400]
$X1$	Distance from 8 to II	[0,L/2]
$X2$	Distance from 9 to II	[max(X1,L/2),L-200]
$X3$	Distance from 12 to II	[X1,min(X2-1300, L/2)]
$X4$	Distance from 13 to II	[X3+1000,X2-200]
Δ	Distance from I to IV	[150,250]

C. 3-D VEHICLE MODELING

3-D vehicle modeling is proposed to generate a different kind of types vehicle model deformed from a 3-D generic model. To project a 3-D vehicle model into the 2-D image frame, the most basic requirement is that the camera is fixed and well-calibrated, (i.e., the 3×4 projection matrix P is known). The 3-D generic model is composed of 16 vertices and 23 arcs, as shown in Figure 4. is defined by 12 shape parameters, including the vehicle length L, vehicle widths, vehicle heights, etc, which are listed in Table 1. According to the definition of vehicle pose in [37], the vehicle's pose is determined by 3 parameters, which are its position (X, Y) on the ground plane and its orientation θ about the vertical axis perpendicular to the ground plane. These total 15 parameters can be calculated in an evolutionary computing framework called estimation of multivariate normal algorithm-global (EMNAglobal) [36], to evaluate the goodness-of-fit between the projection of the 3-D vehicle model and the image data.

A fitness evaluation score(FES) is used as the objective function for evaluating the 3-D vehicle model fitting in an iterative optimization. After projecting the wire-frame 3-D vehicle model onto the image plane, a series of visible line segments have created, and a size of $L \times 2w$ virtual rectangle is assigned for each visible projected line segments l, which is shown in Figure 5. If the projected line segment fits the image edges of interesting vehicle well, the gradient directions of the pixels with a large gradient magnitude value in the rectangle should cluster on the perpendicular direction of this projected line. For each pixel S_i in the virtual rectangle, its gradient magnitude $m(x, y)$ and orientation $\beta(x, y)$ are simply calculated from pixel differences. As a result, the FES E_{S_i} contributed by S_i in the rectangle can be obtained by the vertical component of its gradient magnitude along the line

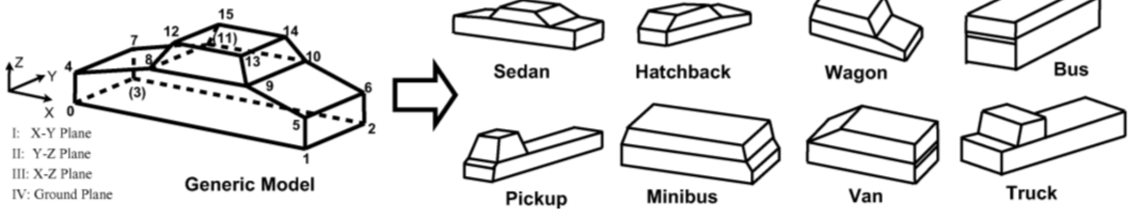


FIGURE 4. Generic model for 3-D modeling and different types of vehicle deformed from generic model.

direction as follows:

$$E_{S_i} = |m(x, y) \cdot \sin(\beta(x, y) - \alpha)| \tag{2}$$

Note that not all pixels within the virtual rectangle have the same contribution to FES, so a standard Gaussian ditribution $G_{(\mu,\sigma)}(d_i)$ with $\mu = 0, \sigma = w$ is further weighted for every pixel S_i, where d_i denotes the distance from S_i to the projected line segment l. Hence, the FES of the visible projected line segment l is calculated by the weighted sum of S_i, and the FES between the projection of the wire-frame 3-D vehicle model and the image data is the sum of all the visible projected line segments.

$$E = \sum_l \left[\log E_l\right] = \sum_l \log\left(\sum_{S_i}\left[E_{S_i} \cdot G_{0,w}(d_i)\right]\right) \tag{3}$$

D. VEHICLE FEATURES MATCHING

After the 3-D model of the vehicle is built, the license plate region of the vehicle can be easily located, and the 12 shape parameters and the color of the vehicle are stored for vehicle re-identification.

To recognize the license plate, the optical character recognition(OCR) based license plate recognition technology is used in this work; however, due to the low video resolution and the movement of vehicle in the surveillance camera, the characters on the license plate are hardly recognized in most of the cases in the *NVIDIA AI City Dataset*, so we adopt the SSD to measure the similarity of the license plate between the previous frame and the coming frames. In the case of completely occlusion, these features of vehicle shape, color, and license plate extracted from the previous frame are used to compare with the newly occurring vehicle in the video. If these features of the vehicle are highly matched in two different frames, the vehicles are considered as identical.

In order to match the vehicle shape features, we estimate the similarity s_{shape} of the 12-D shape parameters between the previous 3-D vehicle model and the subsequent ones. If the similarity s_{shape} is greater than a threshold $\tau_{shape} = 0.6$, the two vehicles are regarded as of the same shape. Moreover, we also estimate the similarity pl_{sim} of the license plate between two different frame, if the pl_{sim} is greater than a threshold $\tau_{sim} = 0.7$, the license plate in two different vehicles are regard as the same. If both the vehicle shape, color, and the license plate are all matched, the vehicle in different frames are considered as the same.

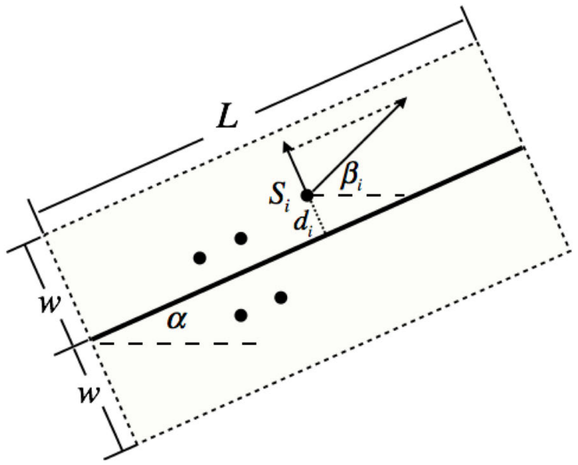


FIGURE 5. Retangular region around a visible projected line *l*.

IV. 3DMCK TRACKING WITH 3D VEHICLE MODEL IN KALMAN FILTER FRAMEWORK

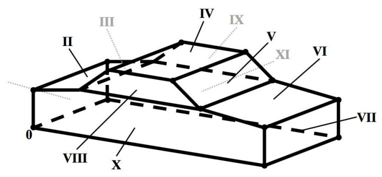
In this section, we mainly describe how to track a vehicle with constrained multiple kernel(CMK) in 3-D space under the framework of the Kalman filter, facilitated with the 3-D vehicle model. In order to integrate the 3-D vehicle's model into 3DCMK tracking, we construct a total of 11 planes for each vehicle and regard each plane of the 3-D vehicle's model as a kernel. Figure 6 shows the generic vehicle model and its built kernels, the corresponding vertices of each kernel annotated by $K\{\cdot\}$. Note that each kernels is a basic component and not all of the kernel are visible in the tracking procedure.

A. MULTIPLE KERNELS IN 3D SPACE

In traditional kernel-based tracking, a histogram including spatial and color information is usually used to represent the target and candidate model. During the histogram extraction, the contribution of a pixel is determined by the distance between the pixel and the kernel center. In [21], the tracking problem for maximizing the similarity $simi(\mathbf{x})$ is formulated as locating $\mathbf{x}$ that maximizes the probability density function (pdf) $f(\mathbf{x})$:

$$f(\mathbf{x}) = \frac{\sum_{i=0}^{N_h} \omega_i k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)}{\sum_{i=0}^{N_h} k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)} \tag{4}$$

where $\mathbf{x}$ is the kernel center; the subscript $\mathbf{i}$ represents each pixel location inside the kernel; $k(\cdot)$ is a kernel function with a convex and monotonic decreasing kernel profile. $\mathbf{z_i}$ and ω_i



K{•}	Vertices	Description
I	0, 3, 4, 7	rear-side
II	4, 7, 8, 11	boot cover
III	8, 11, 12, 15	rear window
IV	12, 13, 14, 15	roof
V	9, 10, 13, 14	windshield
VI	5, 6, 9, 10	engine hood
VII	1, 2, 5, 6	front-side
VIII	8, 9, 12, 13	right window
IX	10, 11, 14, 15	left window
X	0, 1, 4, 5, 8, 9	right-side
XI	2, 3, 6, 7, 10, 11	left-side

FIGURE 6. The vehicle's generic model and the table of kernels built from the 3D vehicle model.

are the position to be considered and the weight of a pixel, respectively; h is the bandwidth of the kernel.

After building a 3-D vehicle model, project it onto the image plane to form a 2-D wire-frame vehicle model, which contains multiple kernels. To describe these kernel in 3-D space, we need to provide the color information to the 3-D kernel. Since the color information from the image is only defined in 2-D space, we associate the color information by back-projecting the 2-D point P^k in the image to the 3-D point $\overleftarrow{P}^k$ on the kernel plane. Therefore, the target can be described by its pdf q in terms of the $r - bin$ histograms,

$$q_u^\kappa = \frac{\sum_{i=1}^{n_k} k\left(\left\|\frac{\mathbf{p}_c^\kappa - \overleftarrow{\mathbf{P}}_i^\kappa}{h}\right\|^2\right)\delta\left[b\left(\mathbf{p}_i^\kappa\right) - u\right]}{\sum_{i=1}^{n_k} k\left(\left\|\frac{\mathbf{p}_c^\kappa - \overleftarrow{\mathbf{P}}_i^\kappa}{h}\right\|^2\right)}, \quad \sum_{u=1}^{r} q_u^\kappa = 1 \tag{5}$$

where $\|\cdot\|$ denotes the L2 norm, the subscript i represents each pixel location inside the kernel, P_c^k is the center and of the kernel in 3-D space; δ is the Kronecker delta function. The function b associates the pixel at location P_i^k with the index of its bin in the color histogram.

During 3DCMK tracking, all candidate kernels search for the region with highest similarity to the target kernel while keeping the predefined constraints unchanged; however, due to the occlusions or view aspects, not all the kernels within the vehicle model are reliable. First, due to the perspective, only a few kernels of the vehicle model projected onto the image are fully visible, only these visible kernels are used for multiple kernel tracking and the other kernels hidden behind are weighted by zero. Second, in order to maintain the interrelationship between the kernels within the 3-D vehicle model, several constraints between the kernels need to be pre-assigned according to the 3-D geometry of the vehicle model. The first constraint is that the distance between $K^3\{k\}$ and $K^3\{k^*\}$ should remain the same initial distance L'_{κ,κ^*}, which implies

$$\left\|\mathbf{p}_c^\kappa - \mathbf{p}_c^{\kappa^*}\right\|^2 = \left(L'_{\kappa,\kappa^*}\right)^2, \quad \text{for visible} \mathrm{K}^3\left\{\kappa \mid \kappa \neq \kappa^*\right\} \tag{6}$$

The second constraint is that the pitch angle ϕ_{k,κ^*} and yaw angle $\varsigma_{\kappa,\kappa^*}$ between $K^3\{k\}$ and $K^3\{k^*\}$ should be keep constant as well. To obtain the constraints for the pitch and yaw, we start to estimate two orthogonal vectors v_a and v_b

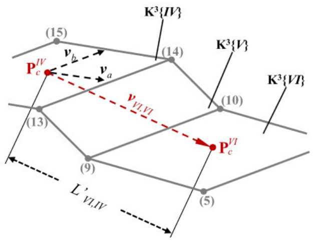


FIGURE 7. Illustration of the constraints for two kernels $K^3\{VI\}$ and $K^3\{IV\}$.

which both across $\mathbf{p}_c^{\kappa^*}$:

$$v_a = \frac{\mathbf{P}_a^k + \mathbf{P}_o^\kappa}{2} - \mathbf{P}_c^{k^*}, \quad v_b = \frac{\mathbf{P}_b^\kappa + \mathbf{P}_o^k}{2} - \mathbf{P}_c^{k^*} \tag{7}$$

where P_o^k is the intersection of two adjacent lines selected from the $K^3\{k\}$. P_a^k and P_b^K are the end points of both wires respectively. Let us define $v_{\kappa,\kappa^*} = \mathbf{P}_c^\kappa - \mathbf{P}_c^{\kappa^*}$, the constraints for pitch and yaw is shown as:

$$\begin{cases} \dfrac{v_a \cdot v_{\kappa,\kappa^*}}{\|v_a\|\|v_{k,\kappa^*}\|} = \cos\left(\phi_{\kappa,\kappa^*}\right) \\ \dfrac{v_b \cdot v_{\kappa,\kappa^*}}{\|v_b\|\|v_{\kappa,\kappa^*}\|} = \cos\left(\varsigma_{\kappa,\kappa^*}\right) \end{cases}, \quad \times \text{ for visible } \mathrm{K}^3\left\{\kappa \mid \kappa \neq \kappa^*\right\} \tag{8}$$

Figure 7 shows an example of the constraints between the kernel $K^3\{VI\}$ and $K^3\{IV\}$, with v_a, v_b, and $v_{VI,IV}$ are all shown in the figure.

B. 3DCMK TARCKING

The objective of 3DCMK tracking is to find the candidate model that has the highest similarity to the target model, which is composed of multiple kernels with pre-specified constraints in 3D space. For an object described by N_k kernels, the total cost function $J(X)$ is defined as the sum of N_k individual kernels cost functions $J_k(X)$, which is inversely proportional to the similarity.

$$J(X) = \sum_{k=1}^{N_k} J_k(X), \quad J_k(X) \propto^1 / \operatorname{simi}_k(X) \tag{9}$$

where $simi_k\,(X)$ is the similarity function at the location $X \in R^3$. In addition, the constraint function $C(X)$ is use to confine the kernels according to their spatial interrelationships, in order to maintain the relative location of each kernel, the constraint function need to be set by $C(X) = 0$. Thus, the problem is further formulated as:

$$\hat{X} = \arg\min_X J(X), \quad \text{subject to } C(X) = 0 \tag{10}$$

Note that after projecting the 3-D vehicle model onto the image, not all of the kernels are visible; moreover, some kernels of the vehicle model cannot be used for matching when occlusion occurs. To overcome this issue, we assign an adaptively adjustable weight w_k for each kernel, so the cost function in (9) can be rewritten as:

$$J(X) = \sum_{k=1}^{N_k} w_k \cdot J_k(X) \tag{11}$$

Although the vehicle can be roughly tracked based on the color information through 3DCMK tracking, the 3-D vehicle model may not fit the image data correctly. To solve this problem, we add the fitness of the 3-D vehicle model into the cost function. In other words, the total cost function becomes:

$$J(X) = \sum_{k=1}^{N_k} w_k \cdot \left(J_k^s(X) + J_k^f(X)\right) \tag{12}$$

where $k = 1, 2, \ldots, N_k$ are the indices of the kernels, $J_k^s(X)$ and $J_k^f(X)$ are the cost function associated with the similarity and the fitness of the k^{th} kernel respectively.

In order to gradually decrease the total cost function and maintain the constraints satisfied during the candidate model searching, the projected gradient method in [38] is used to iteratively solve the constrained optimization problem. The basic concept is to project the movement vector δ_X onto two orthogonal spaces, one is associated with decreasing the total cost function and the other is responsible for satisfying the constraints function $C(X) = 0$. According to the projected gradient method, the projection matrix P which projects the vector onto the space in which the values of the constraint functions remain intact is expressed as $\mathbf{I} - \mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C_x^T}$. After applying P to $-J(X)$, the gradient descent direction, the cost function will decrease. In addition, the $\delta_X^B = -\mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C}(\mathbf{x})$ is introduced to make the constraint functions approach zero when moving along δ_X^B. Therefore, the gradient vector δ_X is described as follow:

$$\begin{aligned}
\delta_\mathbf{x} &= \alpha\left(-\mathbf{I} + \mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C_x^T}\right)\mathbf{W}\left(\mathbf{J_x^s} + \mathbf{J_x^f}\right) \\
&\quad + \left(-\mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C}(\mathbf{x})\right) \\
&= \alpha\left(-\mathbf{I} + \mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C_x^T}\right)\mathbf{W}\mathbf{J_x^s} \\
&\quad + \left(-\mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C}(\mathbf{x})\right) \\
&\quad + \alpha\left(-\mathbf{I} + \mathbf{C_x}\left(\mathbf{C_x^T C_x}\right)^{-1}\mathbf{C_x^T}\right)\mathbf{W}\mathbf{J_x^f} \\
&= \delta_\mathbf{x}^A + \delta_\mathbf{x}^B + \delta_\mathbf{x}^C
\end{aligned} \tag{13}$$

where δ_X is the gradient vector of the $J(X)$; α is the size of searching step; I is a 3×3 identity matrix, $\mathbf{C}(\mathbf{x}) = [c_1(\mathbf{x}) \cdots c_m(\mathbf{x})]^T$ consists of m constraint functions; $C_X \in 3 \times m$ is the gradient matrix of the constraint function with respect to X; $\mathbf{W} = \begin{bmatrix} w_1\mathbf{I} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_{N_k}\mathbf{I} \end{bmatrix}$ and $W_k \propto \mathrm{simi}_k(X)$.

As prove in [27], $\delta_\mathbf{x}^A$ and $\delta_\mathbf{x}^B$ have the following three characteristics. The first one is that $\delta_\mathbf{x}^A$ and $\delta_\mathbf{x}^B$ are orthogonal to each other. The second one is moving along the $\delta_\mathbf{x}^A$ will decreases the total cost function $J(X)$ while keeping same values of the constraint function $C(\mathbf{x})$. The last one is that moving along the $\delta_\mathbf{x}^B$ can lower the absolute values of constraint function $C(\mathbf{x})$. Owing to these three characteristics, the optimal solution can be reached in an iterative manner. The iteration is stopped until either the cost function and the absolute values of constraint are both lower than some given thresholds ε_j and ε_c, respectively, or the iteration count is larger than a threshold T (Algorithm 1 in [27]). On the other hand, $\delta_\mathbf{x}^C$ is the movement vector associated with fitness. Once again, it can be proved that $\delta_\mathbf{x}^C$ is orthogonal to both $\delta_\mathbf{x}^A$ and $\delta_\mathbf{x}^B$ (see Appendix A), which means that moving along $\delta_\mathbf{x}^C$ can achieve a better 3-D vehicle model fitting to the image while maintaining the constraints and similarity of the kernels.

C. KALMAN FILTER PREDICTION AND UPDATE

Kalman filter is a traditional unscented transform-based state estimation method, which is used to approximate the mean and covariance of random variables after a nonlinear conversion. Most of tracking problems can be formulated as a state estimation problem, the tracking target can be regarded as a state, and the tracking problem is to predict and locate where the target(state) will appear in the next time. For this reason, Kalman filter is widely used to solve tracking problems. the traditional Kalman filter is defined as the following:

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{w}_{t-1} \tag{14}$$

$$\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \tag{15}$$

where $\mathbf{x}_t \in R^n$ and $\mathbf{y}_t \in R^m$ denote the state and measurement vector at the time step t, respectively; $\mathbf{F}_t$ is the state transition matrix; $\mathbf{H}_t$ is measurement matrix; $\mathbf{w}_{t-1} \sim N(0, Q)$ and $\mathbf{v}_t \sim N(0, R)$ are the system and measurement noise, these two random variables are uncorrelated Gaussian white-noise sequence, with their covariance matrix Q and R, respectively.

In the stage of prediction, the predictions for state and error covariance are as follows:

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\mathbf{x}_{t-1} \tag{16}$$

$$\hat{\mathbf{P}}_t = \mathbf{F}_t\mathbf{P}_{t-1}\mathbf{F}_t^T + \mathbf{Q}_{t-1} \tag{17}$$

After completing the measurement, the Kalman filter will be updated as follows:

$$\mathbf{K}_t = \hat{\mathbf{P}}_t\mathbf{H}_t^T\left(\mathbf{H}_t\hat{\mathbf{P}}_t\mathbf{H}_t^T + \mathbf{R}_t\right)^{-1} \tag{18}$$

$$\mathbf{x}_t = \hat{\mathbf{x}}_t + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}_t\hat{\mathbf{x}}_t\right) \tag{19}$$

$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\right)\hat{\mathbf{P}}_t \tag{20}$$

The implementation of the Kalman filter algorithm is formulated as follows:

(1) *Initialization:*

For each target, the state vector is defined as $\mathbf{x}_t = \left[u_t\ v_t\ \dot{u}_t\ \dot{v}_t\ a_t\ b_t\right]^T$ and the measurement vector is defined as $\mathbf{y}_t = \left[u_t\ v_t\ a_t\ b_t\right]^T$, where (u_t, v_t), $(\dot{u}_t, \dot{v}_t)$ and (a_t, b_t)

are denote the target position, velocity and size, respectively. Hence, the initial for the state transition matrix $\mathbf{F}_t$ and the measurement matrix $\mathbf{H}_t$ are defined as:

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{H}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

(2) *State transition matrix update:*

The size of vehicle in the image sequence will probably change when it is moving toward or away from the camera, and the extracted color histogram used for similarity measurement is highly dependent on the kernel size. On the other hand, when the multiple kernels tracking is performed on the 3-D vehicle model, the result of segmentation is no longer reliable for estimating the similarity due to the occlusion. hence, the state transition matrix needs to be modified adaptively to reflect the potential size changes, so we embed the factor of vehicle size into the matrix $\mathbf{F}_t$:

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 + \dfrac{\beta \nabla f\left(h_x\right)}{a_{t-1}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 + \dfrac{\beta \nabla f\left(h_y\right)}{b_{t-1}} \end{bmatrix} \tag{22}$$

where β is the step size which also contains the smoothing factor; $\nabla f\,(h)$ is the derivative of the pdf with the kernel bandwidth h. Hence, the predict size of the vehicle becomes to:

$$\begin{bmatrix} \hat{a}_t \\ \hat{b}_t \end{bmatrix} = \begin{bmatrix} a_{t-1} + \beta \nabla f\left(h_x\right) \\ b_{t-1} + \beta \nabla f\left(h_y\right) \end{bmatrix} \tag{23}$$

If the vehicle is occluded so much that the average similarity value of all the kernels is lower than a certain threshold, the mechanism of state transition matrix update stops and $\mathbf{F}_t$ return to the default setting as (21).

(3) *Measurement noise covariance matrix update:*

We use the vehicle tracking result as a measurement to update the Kalman filter during the tracking. Although the system is robust under occlusion by using multiple kernels tracking, it still needs a mechanism to avoid the errors caused by incorrect measurements. It can be seen from (18) and (19) that the Kalman gain $\mathbf{K}_t$ not only controls the tradeoff between using the prediction or the measurement, and but also it is the inversely proportional to the measurement noise covariance matrix R. Hence, we can adaptively adjust the portion measurement contribution to avoid errors by changing the covariance matrix as follows:

$$\mathbf{R} = \begin{bmatrix} \sigma^2 \times J(X) & 0 & 0 & 0 \\ 0 & \sigma^2 \times J(X) & 0 & 0 \\ 0 & 0 & w^2 \times J(X) & 0 \\ 0 & 0 & 0 & h^2 \times J(X) \end{bmatrix} \tag{24}$$

where $J(X)$ is the total cost function of all kernels; σ^2 is the predefined variance value, and w and h are the width and height of the kernel, respectively. With the help of the adaptively covariance matrix, if the total similarity between the candidate and the target vehicle is high, the diagonal term of the covariance matrix will be small. In this way, the Kalman gain will have a larger value, which will make the updated state closer to a reliable measurement.

V. EXPERIMENTAL RESULTS

In this section, we show the experimental results of the proposed system on the NVIDIA AI City Dataset, which are taken with high quality static camera facing at the intersections in urban area in both daytime and night time. The dataset has more than 80 hours of videos in total with the resolution of 1920 × 1280 or 720 × 480. We test two videos within the sub-dataset of silicon valley intersection, which can be obtained from the following website. http://smart-city-sjsu.net/AICityChallenge/data.html. We manually labeled 1,760 locations as ground truth which including 32 vehicles across 1,356 frames, we also test one self-recorded video for license plate recognition. All the experiments are processed on a laptop with an Intel core i7 2.2 GHz CPU with 8G DDR except the vehicle classification part. The implementation is constructed by C/C++, the experimental settings are described as follows: In the 3DCMK tracking, the Kullback-Leibler(K-L) distance is used for all similarity-related measurements, and the histogram of the vehicle is constructed based on the HSV color space with a roof kernel. In the 3-D vehicle model fitting, the parameters in the EMNAglobal algorithm are set as N = 2000, R = 100, and the stopping threshold for the gradient magnitude is setting 2. In the camera calibration, the parameters in the EDA [39] are set as R = 2000, N = 20, $g_{max} = 100$, $\tau_{thresh} = 0.1$, the beginning measurement point on the ground plane is located at(1,0,1), while N_x and N_z are both set as 20, the total 400 points form a 20 × 20 square grid on the 3-D ground plane. In the vehicle feature matching, the values of several thresholds are $\tau_{shape} = 0.6$, $\tau_{ssd} = 0.7$. In addition, in order to improve the robustness of the proposed system, the 3-D vehicle model will automatically rebuild and update every five frames.

A. MULTIPLE VEHICLE TRACKING RESULT

To demonstrate the tracking performance of our proposed system, we compare the performance with the following four different tracking methods: the MAST [40] and MOANA [41] are based on tracking by segmentation, and the other two methods RNN [42] and SORT [43] are based on tracking by detection. To fairly evaluate the tracking

TABLE 2. The tracking performance between different methods.

Methods	MOTA(%)	MOTP(%)	FAF	FP	FN	IDS
3DCMK	82.0	99.5	0.23	7	*310*	0
MAST	*79.8*	91.9	*0.26*	118	214	23
RNN	69.0	96.3	0.40	53	484	8
SORT	61.8	*99.1*	0.50	*13*	629	30
MOANA	73.2	87.9	0.34	164	301	*6*

performance for each method, we adopt the following metrics which are widely used in Multiple Object Tracking(MOT) Challenge [44].

- Multiple Object Tracking Accuracy(MOTA): The measurement of tracking accuracy combines three sources of errors: false positive, false negative and identity switches.
- Multiple Object Tracking Precision(MOTP): The measurement of object localization precision.
- False Alarm per Frame(FAF): The average number of false alarm per frame.
- False Positive(FP): the number of times of system detect an object but the ground truth is not present in the image frame.
- False Negative(FN): the number of times of system failed to detect an object but the ground truth is present in the image frame.
- ID Switches(IDS): the number of times two trajectories switch their IDs.

The comparison of experimental results are shown in Table 2. The proposed method shows the best tracking performance in all metrics except for FN. The reason is that MAST retains more foreground around the object regions to achieve robust tracking; however, extra extracted background information will cause the increase in FP and IDS. The ability of the proposed 3DCMK to deal with occlusion problems can be learned from the fact that there is no identity switching, while the other methods are tending to generate new object identities when occlusion occurs. RNN uses the tracking by detection methods, which can recover most identities when the object is partially occluded. But It cannot continuously update objects under fully occlusion, resulting in low MOTA. MOANA applied the change detection to build more stable background for object extraction, then the cross-matching module is adopted to address the occlusion and object re-identification, resulting low IDS. To facilitate the comparison of experimental results, the red entries in table 2 indicate that the best results in the corresponding columns for each video sequence and blue italics is the second-best.

B. ILLUSTAITION OF VEHICLE MODEL FITTING EVOLUTIONARY PROCEDURE

To illustrate the evolutionary procedure of 3D vehicle model fitting, we select one image frame contains a sliver sedan vehicle from the above experiment. The illustrations of the iteration procedure of vehicle model fitting are shown in Figure 8. As shown in the illustration, all kinds of odd vehicle model are generated in the first generation but, after multiple iterations, the vehicle model fits the image frame better and better, at last, converge to the best shape. Some examples of 3D vehicle models built for different kinds of vehicle types are shown in Figure 9. As we can see that the deformable 3D vehicle model can be transformed from the generic model to Sedan, SUV, Hatchback and Truck, and it fit well with the vehicle in the image frame.

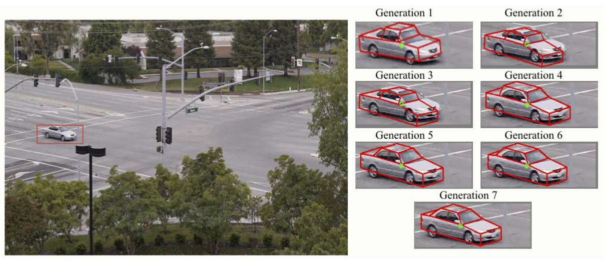


FIGURE 8. 3D vehicle model fitting evolutionary procedure of the sliver sedan.

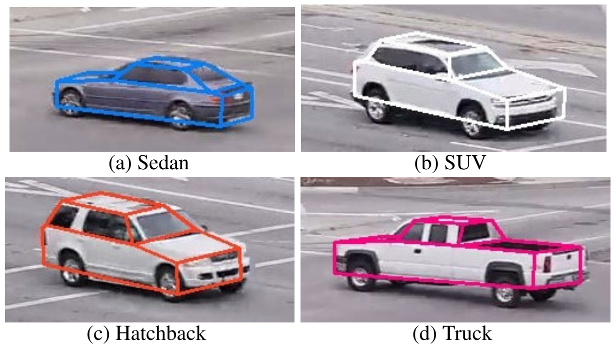
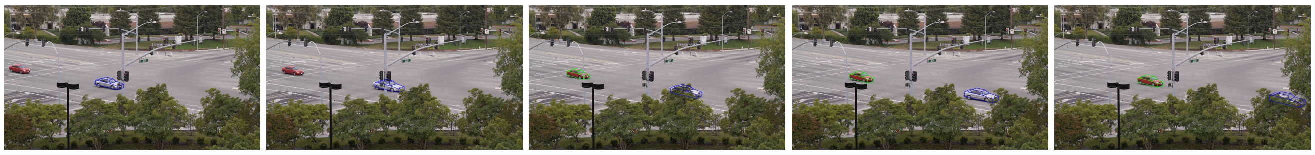


FIGURE 9. Examples of 3D vehicle model built for different kinds of types vehicles.

C. OCCLUSION

In static camera video surveillance, it is common for part of the vehicle to be occluded by other objects in the scene, such as trees and lighting posts at the traffic intersection.

The proposed system can effectively solve the occlusion problem by using 3-D vehicle modeling and 3DCMK tracking technology. On the one hand, the sampling strategy in 3-D vehicle modeling has a strong ability to deal with occlusion. After projecting the 3-D model into the image frame and removing the hidden lines, the sampled points out of the detected bounding box are deleted, and the iterations are completed in the visible region with normalization. The noise generated from the occluded regions would not significantly affect the result of evolutionary procedure. On the other hand, when the 3-D vehicle model is built, 3DCMK tracking is preformed and the occluded kernels in the model are weighted by zero, in this way, the measurement for the Kalman filter update becomes more reliable. At the same time, if the vehicle is completely occluded, the 3-D vehicle model and 3DCMK tracking will become invalid, and the Kalman filter can predict the location of the vehicle in a short period. When the vehicle reappears in the scene, the previously saved vehicle features are used for vehicle re-identification.



(a) Frame #19 (b) Frame #22 (c) Frame #28 (d) Frame #33 (e) Frame #39

FIGURE 10. Results of a Sedan passing by a lighting post and a tree branch.

An example of a sedan passing by a lighting post and a tree branch in the image, and the series of tracking results under partial and complete occlusion is shown in Figure 10. As we can see, in frame 22, the right window and part right-side of the vehicle are occluded by the lighting post, but the 3-D vehicle model still fits the image very well. In frame 28, the vehicle is almost fully occluded by the tree branch, the vehicle is still being tracked because of the Kalman filter and 3DCMK tracking. When the vehicle appears again from frame 33, the feature matching is triggered to re-identify the vehicle. The 3-D vehicle model drifts a little bit in the frame 39, but it still maintains a solid 3-D Sedan structure. In a word, the tracking results under both partial and completely occlusion are overall acceptable, and the 3-D vehicle model fits the image frame as expected.

D. VEHICLE RECOGNITON

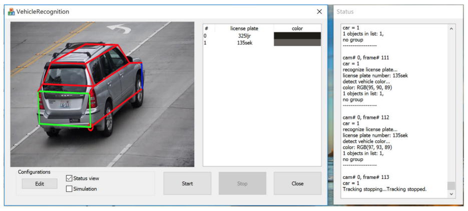
Vehicle recognition and re-identification play an extremely important role in video surveillance systems, and it is also a challenging task due to occlusion, high distortion caused by dynamic viewpoints, and nonuniform illumination conditions.

With the help of a 3-D vehicle model, vehicle recognition becomes quite straightforward. We can easily obtain the model type of vehicle, extract the color information and locate the region of the license plate from the established 3-D vehicle model, then OCR-based LPR is adopted to further recognition if the resolution is high enough.

Due to the perspective, the vehicle in the *NVIDIA AI City Dataset* cannot directly be used for license plate recognition, so we adopt the SSD to measure the similarity between the chopped license plate. We tested 7 different license plates (denoted by 01, 02, . . . , 07) and compared the similarity of the license plate between the initial frame and after its 6^{th} frame(denoted by (01′, 02′ . . . , 07′). The comparison results are shown in Table 3. From the Table 3, It can be seen that the similarity score of each license plate is relatively high when compared with its corresponding ones(red value in the table). We also test one self-recorded video which has sufficient resolution for license plate recognition. An example of using 3D vehicle model for vehicle recognition is shown as Figure 11. As shown in the Figure 11(a), we can intuitively see that the vehicle model is an SUV car, the region of vehicle license plate is inside the green bounding box, and the license plate number and color information of the tracked vehicle are recorded in the middle. More details about the license plate recognition is shown in Figure 11(b). After obtaining the cropped lisence plate region from the 3D vehicle model, we adopt the open-source Tesseract to detect the License plate, then all the characters in the cropped lisence plate image are segmented according to the vertical histogram which can find the gap between license plate characters. Finally, The OCR is applied to analyze each character from the segmented image, and the accuracy of the tested two vehicle license plate characters recognition rate is 66.66%.

TABLE 3. The similarity score of the comparison.

	01	*02*	*03*	*04*	*05*	*06*	*07*
01′	0.772	0.535	0.528	0.546	0.598	0.534	0.618
02′	0.497	0.751	0.484	0.551	0.505	0.491	0.527
03′	0.482	0.458	0.757	0.580	0.457	0.536	0.560
04′	0.561	0.4735	0.500	0.841	0.475	0.523	0.489
05′	0.492	0.499	0.542	0.508	0.760	0.513	0.548
06′	0.566	0.543	0.436	0.484	0.497	0.853	0.439
07′	0.463	0.515	0.558	0.593	0.452	0.517	0.729



#	license plate	color
0	325ljr	⬛
1	135sek	⬛

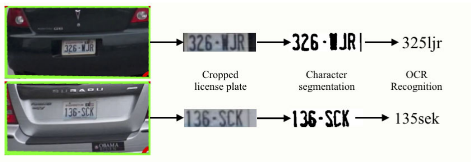
```
car = 1
1 objects in list: 1,
no group
-----------------

cam# 0, frame# 111
car = 1
recognize license plate...
license plate number: 135sek
detect vehicle color...
color: RGB(95, 90, 89)
1 objects in list: 1,
no group
-----------------

cam# 0, frame# 112
car = 1
recognize license plate...
license plate number: 135sek
detect vehicle color...
color: RGB(97, 93, 89)
1 objects in list: 1,
no group
-----------------

cam# 0, frame# 113
car = 1
Tracking stopping...Tracking stopped.
```

(a) vehicle feature extraction



(b) OCR-based License plate recognition

FIGURE 11. Example of vehicle recogniton by using 3D vehicle model.

E. DISCUSSION

The proposed system effectively integrates the 3-D deformable vehicle model fitting, constrained multiple kernels tracking in 3-D space and the Kalman filter framework. With the help of 3DCMK tracking, the partial occlusion issue is handled in the tracking system. Meanwhile, facilitated with the 3D deformable vehicle model, the proposed system cannot only robust track the vehicle, but also can easily extract the vehicle's features(type, color and license plate number) for further vehicle re-identification.

Nevertheless, several limitations are still existed. First, the proposed approach adopts the tracking by detection scheme to detect and then track vehicle, this implies that the approach highly relies on the detection results; besides, if the quality of video sequences is not sufficient for the vehicle detector, the proposed tracking system is not able to perform well on the poor detection results. More specifically, the positive detection of a target can always trigger the tracking of a specific object. In other words, the proposed approach may not work well at night or some cases of insufficient lighting(hazy weather) [45]–[47]. Second, the proposed approach effectively fits the 3D vehicle model into the image frame when the vehicle is tracked, but the vehicle model fitting evolutionary procedure is time-consuming. It usually will take about 1.2 second to build a 3D vehicle model for each vehicle, so the proposed approach is not suitable for real-time vehicle tracking and recognition systems. Last, the proposed method effectively estimates the camera parameters based on the cameras is fixed at a static height, and the vehicle moves on flat roads, but if the camera height is constantly changing will produce less reliable estimation, resulting in larger error of the object back-projection and impacting accuracy of the re-projected 3-D information. Hence, the proposed method is not reliable for the unmanned aerial videos [48]–[50] because its height dynamically changes and then infers unreliable 3-D information of vehicles.

In the future, we will focus on reducing the time cost in the fitting evolutionary procedure. In addition, we will also improve the accuracy of license plate recognition.

VI. CONCLUSION

We propose a robust vehicle tracking and recognition system under a static surveillance camera. The major contribution of this work is three-fold. (1) We propose an innovative 3-D constrained multiple-kernel tracking facilitated with the 3-D vehicle model. Different from previous works of tracking kernels in the 2-D space, the proposed method directly tracks the vehicle in 3-D space. (2) Vehicle feature matching and recognition schemes are designed to overcome the occlusion and ambiguity among neighboring objects, which incorporate both the adaptive appearance model and 3D geometry information. (3) The proposed system also adopts the Kalman filter framework and the RNN-based vehicle detector to achieve a better tracking result. The experimental results of the proposed system shows the favorable performance, which not only can effectively track vehicles under partial and fully occlusion, but also can maintain the knowledge of 3-D vehicle geometry.

APPENDIX

In [30], it is proved that $\delta_{\mathbf{x}}^{A}$ and $\delta_{\mathbf{x}}^{B}$ are orthogonal to each other. The proving that $\delta_{\mathbf{x}}^{C}$ is orthogonal to both $\delta_{\mathbf{x}}^{A}$ and $\delta_{\mathbf{x}}^{B}$ as follows:

$$
\begin{aligned}
&\left(\delta_{\mathbf{x}}^{A}\right)^{T}\left(\delta_{\mathbf{x}}^{C}\right)\\
&=\left(\alpha\left(-\mathbf{I}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\right)\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\\
&\quad\cdot\left(\alpha\left(-\mathbf{I}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\right)\mathbf{J}_{\mathbf{x}}^{f}\right)\\
&=\alpha^{2}\left(-\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}+\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\mathbf{C}_{\mathbf{x}}^{T}\right)\\
&\quad\cdot\left(-\mathbf{J}_{\mathbf{x}}^{f}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\right)\\
&=\alpha^{2}\Bigg(\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\left(\mathbf{J}_{\mathbf{x}}^{f}\right)-\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\\
&\quad-\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\\
&\quad+\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\Bigg)\\
&=\alpha^{2}\Bigg(\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\left(\mathbf{J}_{\mathbf{x}}^{f}\right)-\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\Bigg(\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\\
&\quad+\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}-\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\Bigg)\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\Bigg)\\
&=\alpha^{2}\left(\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\left(\mathbf{J}_{\mathbf{x}}^{f}\right)-\left(\mathbf{J}_{\mathbf{x}}^{s}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\right)\\
&=\alpha^{2}\left(\left(\mathbf{J}_{\mathbf{x}}^{T}\right)^{T}\left(\mathbf{J}_{\mathbf{x}}^{f}\right)-\left(\mathbf{s}_{\mathbf{x}}^{S}\right)^{T}\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}\right)^{-1}\left(\mathbf{C}_{\mathbf{x}}^{T}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{f}\right)\\
&=\alpha^{2}\left(\left(\mathbf{J}_{\mathbf{x}}^{T}\right)\left(\mathbf{J}_{\mathbf{x}}^{f}\right)-\left(\mathbf{J}_{\mathbf{x}}^{T}\left(\mathbf{J}_{\mathbf{x}}^{f}\right)\right)\right)\\
&=0
\end{aligned}
$$

Thus, the $\delta_{\mathbf{x}}^{C}$ is orthogonal to $\delta_{\mathbf{x}}^{A}$.

$$
\begin{aligned}
&\left(\delta_{\mathbf{x}}^{B}\right)^{T}\left(\delta_{\mathbf{x}}^{C}\right)\\
&\quad=\left(-\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}(\mathbf{x})\right)^{T}\\
&\qquad\times\left(\alpha\left(-\mathbf{I}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\right)\mathbf{J}_{\mathbf{x}}^{s}\right)\\
&\quad=\alpha\left(-\mathbf{C}(\mathbf{x})^{T}\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}\mathbf{C}_{\mathbf{x}}^{T}\right)\\
&\qquad\times\left(-\mathbf{J}_{\mathbf{x}}^{s}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{s}\right)\\
&\quad=\alpha\left(\mathbf{C}(\mathbf{x})^{T}\mathbf{C}_{\mathbf{x}}^{-1}\mathbf{J}_{\mathbf{x}}^{s}-\mathbf{C}(\mathbf{x})^{T}\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{s}\right)\\
&\quad=\alpha\mathbf{C}(\mathbf{x})^{T}\left(\left(\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)^{T}-\left(\mathbf{C}_{\mathbf{x}}^{T}\mathbf{C}_{\mathbf{x}}\right)^{-1}\right)\mathbf{C}_{\mathbf{x}}^{T}\mathbf{J}_{\mathbf{x}}^{s}\\
&\quad=0
\end{aligned}
$$

Therefore, the $\delta_{\mathbf{x}}^{C}$ is orthogonal to $\delta_{\mathbf{x}}^{B}$.

REFERENCES

[1] D. Koller, J. Weber, and J. Malik, "Towards realtime visual based tracking in cluttered traffic scenes," in *Proc. Intell. Veh. Symp.*, 1994, pp. 201–206.

[2] E. B. Meier and F. Ade, "Tracking cars in range images using the condensation algorithm," in *Proc. IEEE/IEEJ/JSAI Int. Conf. Intell. Transp. Syst.*, Oct. 1999, pp. 129–134.

[3] P. L. M. Bouttefroy, A. Bouzerdoum, S. L. Phung, and A. Beghdadi, "Vehicle tracking using projective particle filter," in *Proc. 6th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2009, pp. 7–12.

[4] J. Scharcanski, A. B. de Oliveira, P. G. Cavalcanti, and Y. Yari, "A particle-filtering approach for vehicular tracking adaptive to occlusions," *IEEE Trans. Veh. Technol.*, vol. 60, no. 2, pp. 381–389, Feb. 2011.

[5] J. Lou, T. Tan, W. Hu, H. Yang, and S. J. Maybank, "3-D model-based vehicle tracking," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1561–1569, Oct. 2005.

[6] Z. Zhang, K. Huang, T. Tan, and Y. Wang, "3D model based vehicle tracking using gradient based fitness evaluation under particle filter framework," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 1771–1774.

[7] Z. Zhang, T. Tan, K. Huang, and Y. Wang, "Three-dimensional deformable-model-based localization and recognition of road vehicles," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 1–13, Jan. 2012.

[8] M. J. Leotta and J. L. Mundy, "Investigating 3-D model and part information for improving content-based vehicle retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 7, pp. 1457–1469, Jun. 2010.

[9] Y. Li, L. Gu, and T. Kanade, "Robustly aligning a shape model and its application to car alignment of unknown pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1860–1876, Sep. 2011.

[10] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[11] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 920–939, Sep. 2011.

[12] P. L. M. Bouttefroy, A. Bouzerdoum, S. L. Phung, and A. Beghdadi, "Vehicle tracking by non-drifting mean-shift using projective Kalman filter," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 61–66.

[13] W. Zhang, Q. M. J. Wu, X. Yang, and X. Fang, "Multilevel framework to detect and handle vehicle occlusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 161–174, Mar. 2008.

[14] Y. Wang, "Real-time moving vehicle detection with cast shadow removal in video based on conditional random field," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 437–441, Mar. 2009.

[15] T. Xiong and C. Debrunner, "Stochastic car tracking with line- and color-based features," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 324–328, Dec. 2004.

[16] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 175–187, Jun. 2006.

[17] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 259–289, May 2008.

[18] J.-Y. Choi, K.-S. Sung, and Y.-K. Yang, "Multiple vehicles detection and tracking based on scale-invariant feature transform," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2007, pp. 528–533.

[19] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. Eur. Conf. Comput. Vis.* IEEE, 2002, pp. 661–675.

[20] R. Guerrero-Gómez-Olmedo, R. J. López-Sastre, S. Maldonado-Bascón, and A. and Fernández-Caballero, "Vehicle tracking by simultaneous detection and viewpoint estimation," in *Proc. Int. Work-Conf. Interplay Between Natural Artif. Comput.* IEEE, 2013, pp. 306–316.

[21] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[22] C. Yang, R. Duraiswami, and L. Davis, "Efficient mean-shift tracking via a new similarity measure," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 176–183.

[23] A. Yilmaz, "Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–6.

[24] T. Liu, Y. Liu, Z. Tang, and J.-N. Hwang, "Adaptive ground plane estimation for moving camera-based 3D object tracking," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSP)*, 2017, pp. 1–6.

[25] Z. Fan, Y. Wu, and M. Yang, "Multiple collaborative kernel tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2005, pp. 502–509.

[26] J. Fang, J. Yang, and H. Liu, "Efficient and robust fragments-based multiple kernels tracking," *AEU-Int. J. Electron. Commun.*, vol. 65, no. 11, pp. 915–923, Nov. 2011.

[27] C.-T. Chu, J.-N. Hwang, H.-I. Pai, and K.-M. Lan, "Tracking human under occlusion based on adaptive multiple kernels with projected gradients," *IEEE Trans. Multimedia*, vol. 15, no. 7, pp. 1602–1615, Nov. 2013.

[28] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences," *Int. J. Comput. Vis.*, vol. 35, no. 3, pp. 295–319, Dec. 1999.

[29] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker, "A generic deformable model for vehicle recognition," in *Proc. BMVC*, vol. 1, 1995, p. 2.

[30] T. N. Tan and K. D. Baker, "Efficient image gradient based vehicle localization," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1343–1356, Aug. 2000.

[31] Z. Zhang, M. Li, K. Huang, and T. Tan, "3D model based vehicle localization by optimizing local gradient based fitness evaluation," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.

[32] Y. Zheng and S. Peng, "Model based vehicle localization for urban traffic surveillance using image gradient based matching," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 945–950.

[33] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[34] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[35] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *Int. J. Comput. Vis.*, vol. 4, no. 2, pp. 127–139, Mar. 1990.

[36] P. Larra naga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, vol. 2. CoRR, 2001.

[37] T.-N. Tan, G. D. Sullivan, and K. D. Baker, "Model-based localisation and recognition of road vehicles," *Int. J. Comput. Vis.*, vol. 27, no. 1, pp. 5–25, 1998.

[38] J. A. Snyman, *Practical Mathematical Optimization*. Springer, 2005.

[39] Z. Tang, Y.-S. Lin, K.-H. Lee, J.-N. Hwang, J.-H. Chuang, and Z. Fang, "Camera self-calibration from tracking of moving persons," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 265–270.

[40] Z. Tang, J.-N. Hwang, Y.-S. Lin, and J.-H. Chuang, "Multiple-kernel adaptive segmentation and tracking (MAST) for robust object tracking," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 1115–1119.

[41] Z. Tang and J.-N. Hwang, "MOANA: An online learned adaptive appearance model for robust multiple object tracking in 3D," *IEEE Access*, vol. 7, pp. 31934–31945, 2019.

[42] G. P. Mauroy and E. W. Kamen, "Multiple target tracking using recurrent neural networks," in *Proc. Int. Conf. Neural Netw. (ICNN)*, Jul. 2017, pp. 2076–2079.

[43] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.

[44] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016, *arXiv:1603.00831*. [Online]. Available: http://arxiv.org/abs/1603.00831

[45] G. Li, Y. Yang, X. Qu, D. Cao, and K. Li, "A deep learning based image enhancement approach for autonomous driving at night," *Knowl.-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106617.

[46] G. Li, Y. Yang, and X. Qu, "Deep learning approaches on pedestrian detection in hazy weather," *IEEE Trans. Ind. Electron.*, vol. 67, no. 10, pp. 8889–8899, Oct. 2020.

[47] G. Li, Y. Lin, and X. Qu, "An infrared and visible image fusion method based on multi-scale transformation and norm optimization," *Inf. Fusion*, vol. 71, pp. 109–129, Jul. 2021.

[48] X. Chen, Z. Li, Y. Yang, L. Qi, and R. Ke, "High-resolution vehicle trajectory extraction and denoising from aerial videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 3190–3202, May 2021.

[49] D. Jiang, B. Wu, Z. Cheng, J. Xue, and P. H. A. J. M. van Gelder, "Towards a probabilistic model for estimation of grounding accidents in fluctuating backwater zone of the three gorges reservoir," *Rel. Eng. Syst. Saf.*, vol. 205, Jan. 2021, Art. no. 107239.

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