applied sciences

MDPI

Article

Multi-Vehicle Tracking Based on Monocular Camera in Driver View

Pengfei Lyu [1], Minxiang Wei [1,*] and Yuwei Wu [2]

[1] College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[2] College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
* Correspondence: weimx@nuaa.edu.cn

Abstract: Multi-vehicle tracking is used in advanced driver assistance systems to track obstacles, which is fundamental for high-level tasks. It requires real-time performance while dealing with object illumination variations and deformations. To this end, we propose a novel multi-vehicle tracking algorithm based on a monocular camera in driver view. It follows the tracking-by-detection paradigm and integrates detection and appearance descriptors into a single network. The one-stage detection approach consists of a backbone, a modified BiFPN as a neck layer, and three prediction heads. The data association consists of a two-step matching strategy together with a Kalman filter. Experimental results demonstrate that the proposed approach outperforms state-of-the-art algorithms. It is also able to solve the tracking problem in driving scenarios while maintaining 16 FPS on the test dataset.

Keywords: multi-vehicle tracking; object detection; data association; Kalman filter

Citation: Lyu, P.; Wei, M.; Wu, Y. Multi-Vehicle Tracking Based on Monocular Camera in Driver View. *Appl. Sci.* **2022**, *12*, 12244. https://doi.org/10.3390/app122312244

Academic Editor: Emanuele Carpanzano

Received: 24 October 2022
Accepted: 25 November 2022
Published: 30 November 2022

1. Introduction

Cameras are an attractive sensor choice for intelligent vehicles because they are passive and easy to deploy. Some existing driving assistance systems rely on cameras to detect pedestrians [1] or for lane keeping [2]. Meanwhile, as a research hot-spot in the field of computer vision, the objective of multi-target tracking is to estimate the trajectories of a specific class of objects, such as vehicles [3,4]. Furthermore, multi-vehicle tracking based on a monocular camera plays a vital role in advanced driver assistance systems.

Currently, there are two main target tracking paradigms: detection-free-tracking (DFT) and tracking-by-detection (TBD) [5]. DFT usually initializes the tracking target manually, which is suitable for tracking a specified target. However, it cannot track a new object that appears in a driving scene. TBD means tracking-by-detection, which can automatically detect the appearance of new targets or the disappearance of existing targets. Therefore, TBD can meet actual demand for the random disappearance or dynamical change of targets in driving scenes. There are various TBD approaches in multi-target tracking development. For instance, DeepSORT [6] uses two-stage processing and achieves pretty good performance. It first obtains bounding boxes of detected objects through the detection network and then uses the pre-trained deep convolution neural network to perform the appearance description. Recently, the single-shot TBD method [7] integrates the detection network and the appearance descriptor into a single framework which avoids re-computation and simultaneously outputs the embedding information and detected bounding box. This kind of framework is suitable for advanced driver assistance systems, which are time-critical applications.

As one of the basic problems of computer vision, target detection is the foundation of many other visual tasks, such as target tracking. To a certain extent, the quality of the target detection module determines the performance of the multi-target tracking performance based on the TBD paradigm. The object detection method includes one-stage and two-stage

detection algorithms. The Faster-RCNN [8] is a representative work of a two-stage detector. On the contrary, the YOLOv3 [9] is a classic one-stage object detection approach, which ignores the region proposal stage and replaces it by a single network to directly output the predicted bounding boxes and classification scores in the image. Although two-stage detection algorithms are generally more accurate than the one-stage detection approaches when detecting small targets, the one-stage detectors are faster. In this way, a one-stage detector is useful for vehicles detection whose size is generally larger within a safe distance between the ego-vehicle and surrounding vehicles.

Data association (DA) is the other part of the TBD approach. Its objective is to match detection results between adjacent frames of the video stream based on some metrics, such as the cosine distance and Intersection-over-Union (IoU) distance. To improve matching accuracy, the motion state prediction and linear assignment of the object are studied by some studies in the literature. Kalman filter [10] is an important method to solve the motion model, which includes the prediction and updating step. First, it predicts the object state based on the system state space model. Then, it corrects the prediction using the received measurements. Moreover, the optimal Hungarian algorithm [11] is normally employed to solve the linear assignment problem. Each object in the frame is assigned a Kalman filter. Thus, the number of Kalman filters and the dimension of the cost matrix which is fed into the Hungarian algorithm increase as the number of tracking objects increases.

In this work, we focus on multiple vehicle tracking problems. The proposed method only uses the current and previous images that come from a monocular camera in driver view. It can track multiple vehicles in as real time as possible based on the single-shot TBD method. Here are our contributions:

- Following the TBD paradigm, we design a detector, which is composed of a backbone block, a neck layer, and three prediction heads, to enhance the multi-vehicle tracking (MVT) performance;
- For data association, we derive the Kalman filter that is used to update the tracks, which decreases the identity switch to improve the HOTA and other relevant metrics;
- Quantitative and qualitative evaluations demonstrate that our method achieves good performance and can solve the multiple vehicle-tracking problems in the driver view with monocular cameras (e.g., illumination variants and deformations) while maintaining a good frame rate.

The paper is structured as follows: Section 2 mainly introduces the existing related research work. Section 3 presents the general structure of the proposed algorithm and the derivation of Kalman filtering. Section 4 introduces the implementation details, including the acquisition of anchor setting, training and testing details of the algorithm. Section 5 shows the experimental results compared with state-of-the-art algorithms on public datasets. Section 6 presents the conclusions and outlook of the follow-up work.

2. Related Work

System latency is a major concern in autonomous vehicles. However, existing feature extractors are usually time-consuming. In addition, existing MOT frameworks usually deal with detection and association separately, resulting in the accumulation of delays and errors. To avoid such problems, the ability to use a shared neural network for detection and tracking may reduce latency and enhance accuracy.

Assuming that the object moves slightly between consecutive frames, the Tracktor [12] uses an object detector and converts it into a tracker. It adapts to the Faster R-CNN detector [8] by adding a regression head, which regresses the position of the bounding box of the object in the new frame from the previous frame. However, there are two disadvantages. First of all, it is suitable for high frame-rate videos with less inter-frame motion. Secondly, it fails to capture the appearance of tracked objects, and when several objects are close together and under temporary occlusion, the model becomes unreliable. To solve the second problem, Tractor++ came into being. It integrates appearance features

generated by a separate Siamese network, and it achieves better tracking accuracy at the cost of efficiency. However, this part offsets the advantage of Tracktor.

To obtain better long-term correlation, Wang first proposed a synchronous detection and tracking method based on appearance features called JDE [7]. It uses a unified network to combine output detections with their corresponding appearance embedding. JDE extends YOLOv3 [9] with a re-identification branch that is trained using the triplet loss and jointly learned with the detection loss. For each new frame, JDE uses the combination of all embedding of detections belonging to the trajectory wave to update the appearance embedding of the trajectory wave. The similarity score is a combination of the cosine distance between the appearance embedding and the Mahalanobis distance between the current detection and the predicted bounding box of the trajectory wave computed using the Kalman filter. Then, the Hungarian algorithm is used to find the optimal allocation. JDE speeds up the reasoning time. However, it is only evaluated on the pedestrian tracking dataset. Meanwhile, the detection module and Kalman filter can be further improved.

These multi-object tracking algorithms always focus on pedestrian tracking in monitor view. Few such works pay attention to multi-vehicle tracking from the driver's point of view. JDE methods show better results and take less running time than the non-joint learning method. Therefore, we propose a new detection structure based on the idea of JDE and improve the update strategy of the noise matrix. The proposed method can be used as the basis of autonomous driving or ADAS.

3. Proposed Algorithm

3.1. Over View of Tracking Framework

As shown in Figure 1, the proposed tracking method is composed of the joint detection and embedding (JDE) [7] module and the data association (DA) module. JDE is based on a one-stage object detection network which simultaneously outputs the detection results and the appearances of detected targets. DA is a post process which is used to match the detection result from the JDE module with the tracking information from the previous frame.

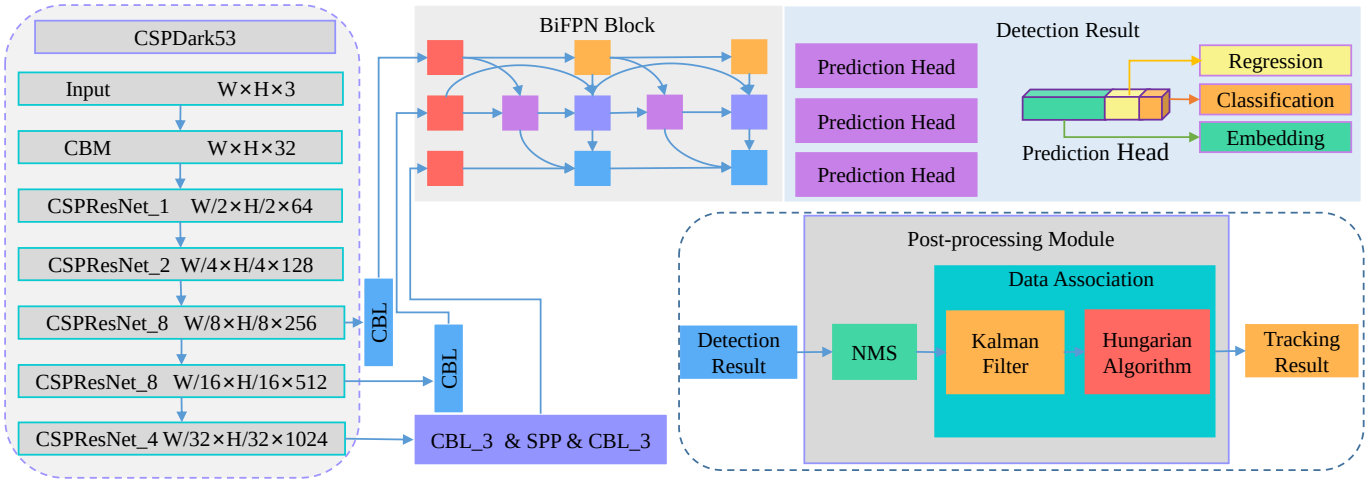

Figure 1. The whole structure of the algorithm consists of a detection module and a post-processing module.

3.2. JDE Module

There are three prediction heads at the end of our designed one-stage detection architecture. The detection part combines a CSPDarkNet-53 as backbone and a modified bi-directional feature pyramid network (BiFPN) as the neck layer [13].

CSPDarkNet-53 is an upgraded version of the DarkNet-53 [9] which is a feature extractor used for object detection. Its core unit is the CSPResNet, which divides a basic feature map into two parts by using CSPNet strategy, and then merges these parts through

a cross-stage hierarchical structure. In the way, this convolutional neural network is used as a backbone in YOLOv4 [14] and transferred to our JDE without any modification.

The BiFPN is a multi-scale feature fusion block which balances efficiency and accuracy. It fuses three feature maps that are output by the backbone block and enables features to flow in both the top–down and bottom–up directions. Different from traditional methods, BiFPN can learn the importance of the input feature maps which have different resolutions by adding an additional weight. Then, these fused features are fed into the corresponding prediction head, which has some convolution layers. As shown in Figure 1, the proposed architecture combines two BiFPN blocks. What is more, we can extend the framework more deeply and widely by increasing the number of BiFPN blocks as well as its input features.

Each prediction head is composed of a series of convolution layers and outputs a three-dimension map with size $(4A + 2A + E) \times H \times W$, where A is the number of anchors that are assigned to its corresponding feature scale, and E is the number of appearance embedding. The three-dimension map consists of the box coordinate offsets regression of size $4A \times H \times W$, the box classification of size $2A \times H \times W$, and the appearance embedding of size $E \times H \times W$.

3.3. Loss Function

There are three subtasks of each prediction head in our multi-vehicle tracker, so the total loss function $\mathcal{L}$ consists of classification loss, regression loss, and embedding loss of all prediction heads,

$$\mathcal{L} = \sum_{i=1}^{N} (w_{cls}^{i} \mathcal{L}_{cls}^{i} + w_{reg}^{i} \mathcal{L}_{reg}^{i} + w_{emb}^{i} \mathcal{L}_{emb}^{i}) \tag{1}$$

where $N = 3$ is the number of prediction heads, w_{cls}^{i} and $\mathcal{L}_{cls}^{i}$ are the weight and loss function for the foreground/background classification task, w_{reg}^{i} and $\mathcal{L}_{reg}^{i}$ are the weight and loss function for the bounding box regression task, w_{emb}^{i} and $\mathcal{L}_{emb}^{i}$ are the weight and loss function for the appearance embedding task, and $i = 1, 2, 3$. All w_{*}^{i} are carefully tuned for optimal performance.

In addition, $\mathcal{L}_{cls}$ is formulated as a cross-entropy loss as below:

$$\mathcal{L}_{cls} = -(y \log(p) + (1 - y) log(1 - p)) \tag{2}$$

where y is a binary indicator (0 or 1) for foreground and background classification, and p is a probability which predicted as positive class.

When it comes to regression loss, we use smooth $L1$ loss:

$$\mathcal{L}_{reg} = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & otherwise. \end{cases} \tag{3}$$

Moreover, $\mathcal{L}_{emb}$ is formulated as the same in [7] as below:

$$\mathcal{L}_{emb} = -\log \frac{\exp(f^T g^+)}{\exp(f^T g^+) + \sum_i (\exp(f^T g_i^-))} \tag{4}$$

where f^T is a selected anchor instance in a mini-batch, g^+ is the weight of the positive sample with respect to f^T, and g_i^- is the weight of the negative sample.

Let m and v denote the 1st moment vector and 2nd moment vector, respectively. α is the learning rate of the model. β_1, β_2 and ε are hyper-parameters. The parameters m, v, and θ are updated during training as follows:

$$\begin{cases} m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\dfrac{\partial Loss}{\partial \theta_{t-1}}, \\ v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\dfrac{\partial Loss}{\partial \theta_{t-1}}^2, \\ \hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t}, \\ \hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t}, \\ \theta_t \leftarrow \theta_{t-1} - \alpha \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}. \end{cases} \tag{5}$$

3.4. Data Association Module

The DA is the other part of the TBD paradigm. After the detection results are acquired, the non-maximum suppression (NMS) algorithm is used to find out the best bounding boxes in these detection results.

As demonstrated in Figure 2, the data association algorithm has two match steps, an embedding match and an IoU match, and a Kalman filter. First of all, the current frame detections and the last frame tracks are matched by an embedding match, which outputs matched tracks, unmatched tracks, and unmatched detections. Then, the last two outputs are matched again with the IoU match. It also has three same format outputs. All of the matched tracks, as well as matched detections, are fed into the Kalman filter. The unmatched tracks will be deleted after 30 frames if they are unmatched in the future matched cycles. The unmatched current frame detections are created as new tracks and put in the inactive tracking pool. The Kalman filter and the matching work together to produce an active tracking pool that outputs the final object bounding boxes with tracked IDs if the condition is met.

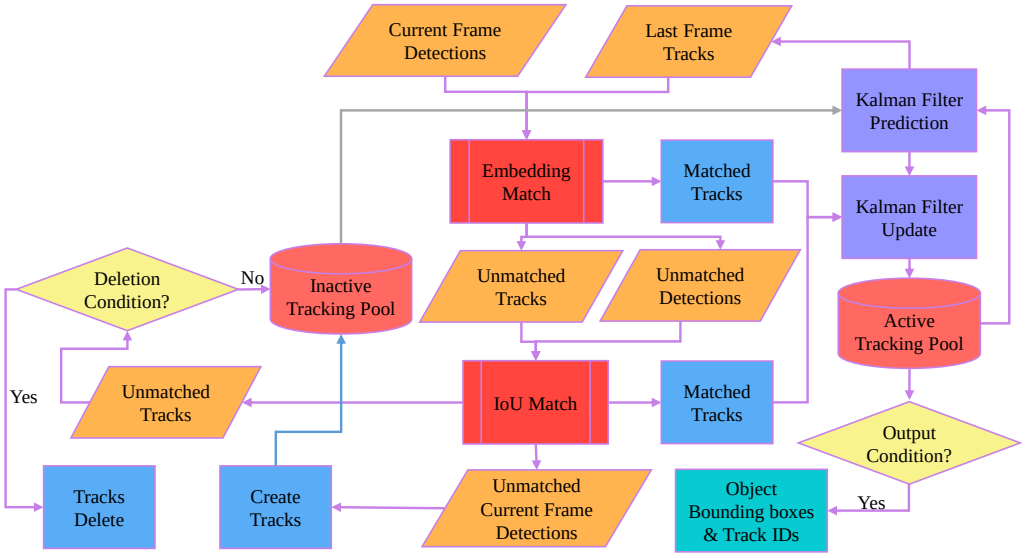

Figure 2. The pipeline of the data association algorithm that has two match steps and a Kalman filter.

3.5. Motion Model

In multiple vehicle tracking, we use the constant velocity as a motion model with a Kalman filter [15] when we assume the tracking system as a linear Gaussian process. In the following section, we derive the constant velocity motion model for objects represented with

aspect ratio and height as well as the detected bounding box-center coordinate. Suppose each target follows a linear Gaussian model:

$$f_{k|k-1}(x|\epsilon) = \mathcal{N}(x; F_{k-1}\epsilon, Q_{k-1}) \tag{6}$$

$$g_k(z|x) = \mathcal{N}(z; H_k x, R_k) \tag{7}$$

where $f_{k|k-1}(\cdot|\epsilon)$ is the state transition probability of a single object at time k when given the previous state ϵ. $g_k(z|x)$ is the likelihood function of the single object, which defines the probability that z is observed under conditions as state x. $\mathcal{N}(\cdot; m, P)$ is a Gaussian density whose mean and covariance are m and P, respectively. F_{k-1} is the state transition matrix and Q_{k-1} denotes the covariance matrix of the process noises. H_k is the measurement matrix. R_k denotes the covariance matrix of the measurement noises, which can be measured from the detection and ground truth of training datasets.

Our objective now is to obtain the formulation of F_k, Q_k, H_k, and R_k. Suppose that the centers of the detection box to be estimated are denoted by $(x_{b,k}, y_{b,k})$ and that the aspect ratio and the height of the bounding box detected at the coordinates of the image to be estimated are represented by $a_{b,k}$ and $h_{b,k}$, respectively, at time k. The velocity of the centers of the detected box, the bounding box, the aspect ratio, and the height are indicated by $(\dot{x}_{b,k}, \dot{y}_{b,k})$, $\dot{a}_{b,k}$, and $\dot{h}_{b,k}$, respectively. The state space model can also be expressed as a vector–matrix representation as follows:

$$\underbrace{\begin{bmatrix} x_{b,k} \\ y_{b,k} \\ \dot{x}_{b,k} \\ \dot{y}_{b,k} \\ a_{b,k} \\ h_{b,k} \\ \dot{a}_{b,k} \\ \dot{h}_{b,k} \end{bmatrix}}_{X_k} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{F_{k-1}} \underbrace{\begin{bmatrix} x_{b,k-1} \\ y_{b,k-1} \\ \dot{x}_{b,k-1} \\ \dot{y}_{b,k-1} \\ a_{b,k-1} \\ h_{b,k-1} \\ \dot{a}_{b,k-1} \\ \dot{h}_{b,k-1} \end{bmatrix}}_{X_{k-1}} + \underbrace{\begin{bmatrix} \frac{\delta t^2}{2} w_{x,k-1} \\ \frac{\delta t^2}{2} w_{y,k-1} \\ \delta t w_{x,k-1} \\ \delta t w_{y,k-1} \\ \frac{\delta t^2}{2} w_{a,k-1} \\ \frac{\delta t^2}{2} w_{h,k-1} \\ \delta t w_{a,k-1} \\ \delta t w_{h,k-1} \end{bmatrix}}_{W_{k-1}} \tag{8}$$

where $w_{*,k-1}$ denotes a piece-wise constant white acceleration that can be described by a zero-mean Gaussian white noise as $w_{*,k-1} \sim \mathcal{N}(0, \sigma^2_{*,k-1})$. The $\sigma^2_{*,k-1}$ is a variance which determines the relaxation level of the constant velocity assumption. δt is the time between frames. Equation (8) can be also represented as:

$$X_k = F_{k-1} X_{k-1} + W_{k-1} \tag{9}$$

where the value of the state transition matrix F_{k-1} is given in Equation (8) and $W_{k-1} \sim \mathcal{N}(0, Q_{k-1})$. Therefore, Q_{k-1} can be obtained by computing the covariance of W_{k-1} as:

$$Q_{k-1} = Cov(W_{k-1}) = E[W_{k-1} W_{k-1}^T] \tag{10}$$

where $E[W_{k-1} W_{k-1}^T]$ denotes the mean of W_{k-1} and W_{k-1}^T. Now Q_{k-1} can be obtained as:

$$Q_{k-1} = \begin{bmatrix} \frac{\delta t^4}{4}\sigma^2_{w_x} & 0 & \frac{\delta t^3}{2}\sigma^2_{w_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\delta t^4}{4}\sigma^2_{w_y} & 0 & \frac{\delta t^3}{2}\sigma^2_{w_y} & 0 & 0 & 0 & 0 \\ \frac{\delta t^3}{2}\sigma^2_{w_x} & 0 & \delta t^2\sigma^2_{w_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\delta t^3}{2}\sigma^2_{w_y} & 0 & \delta t^2\sigma^2_{w_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\delta t^4}{4}\sigma^2_{w_a} & 0 & \frac{\delta t^3}{2}\sigma^2_{w_a} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\delta t^4}{4}\sigma^2_{w_h} & 0 & \frac{\delta t^3}{2}\sigma^2_{w_h} \\ 0 & 0 & 0 & 0 & \frac{\delta t^3}{2}\sigma^2_{w_a} & 0 & \delta t^2\sigma^2_{w_a} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\delta t^3}{2}\sigma^2_{w_h} & 0 & \delta t^2\sigma^2_{w_h} \end{bmatrix} \tag{11}$$

There are two significant ideas in the derivation of Q_{k-1}:

- $E[w_{x,k-1}w_{x,k-1}] = \sigma^2_{w_x}$. Meanwhile, $E[w_{y,k-1}w_{y,k-1}] = \sigma^2_{w_y}$, $E[w_{a,k-1}w_{a,k-1}] = \sigma^2_{w_a}$, and $E[w_{h,k-1}w_{h,k-1}] = \sigma^2_{w_h}$ where $\sigma^2_{w_x}$ is variance ($\sigma_{w_x} = \sqrt{\sigma^2_{w_x}}$ is the standard deviation).
- $E[w_{x,k-1}w_{y,k-1}] = 0$ since between the x-axis and y-axis, there is no correlation. Similarly, $E[w_{x,k-1}w_{a,k-1}] = 0$, $E[w_{a,k-1}w_{h,k-1}] = 0$, etc.

This completes the derivation for F_{k-1} and Q_{k-1}. We set the different values for the variance ($\sigma^2_{w_x}$, $\sigma^2_{w_y}$, $\sigma^2_{w_a}$, and $\sigma^2_{w_h}$) after tuning them individually during our experiments.

To speed up the operation, the observations at time k can be represented by the following state space model in vector matrix form.

$$\underbrace{\begin{bmatrix} z_{x,k} \\ z_{y,k} \\ z_{a,k} \\ z_{h,k} \end{bmatrix}}_{Z_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{H_k} \underbrace{\begin{bmatrix} x_{b,k} \\ y_{b,k} \\ \dot{x}_{b,k} \\ \dot{y}_{b,k} \\ a_{b,k} \\ h_{b,k} \\ \dot{a}_{b,k} \\ \dot{h}_{b,k} \end{bmatrix}}_{X_k} + \underbrace{\begin{bmatrix} v_{x,k} \\ v_{y,k} \\ v_{a,k} \\ v_{h,k} \end{bmatrix}}_{V_k} \tag{12}$$

where $(z_{x,k}, z_{y,k})$ are the center points of a detection box at time k. $z_{a,k}$ and $z_{h,k}$ are the aspect ratio and height of a detection box in image coordinates at time k. $v_{x,k}$, $v_{y,k}$, $v_{a,k}$, and $v_{h,k}$ are observation noises corresponding to $z_{x,k}$, $z_{y,k}$, $z_{a,k}$, and $z_{h,k}$, respectively, which are basically zero-mean Gaussian white noises. For instance, $v_{x,k} \sim \mathcal{N}(0, \sigma^2_{z_{x,k}})$.

The measurement matrix H_k projects the state space to the observation space and Equation (12) can also be represented as

$$Z_k = H_k X_k + V_k \tag{13}$$

where the value of H_k is given in Equation (12) and $V_k \sim \mathcal{N}(0, R_k)$. Thus, the value of R_k can be obtained by computing the covariance of V_k as:

$$R_k = Cov(V_k) = E[V_k V_k^T] \tag{14}$$

Here, the covariance matrix of the measurement noise R_k is expressed as:

$$R_{k-1} = \begin{bmatrix} \sigma^2_{v_x} & 0 & 0 & 0 \\ 0 & \sigma^2_{v_y} & 0 & 0 \\ 0 & 0 & \sigma^2_{v_a} & 0 \\ 0 & 0 & 0 & \sigma^2_{v_h} \end{bmatrix} \tag{15}$$

where $E[v_{x,k}v_{a,k}] = 0$, $E[v_{x,k}v_{y,k}] = 0$, $E[v_{a,k}v_{h,k}] = 0$ and so on, since there is not any correlation between each other. This completes the derivation for H_k and R_k. We set the different values for the variance ($\sigma^2_{v_x}$, $\sigma^2_{v_y}$, $\sigma^2_{v_a}$, and $\sigma^2_{v_h}$) after tuning them individually during our experiments.

Then, every track is assigned a filter and follows the prediction and update process to acquire an estimated state. To some extent, the speed of the tracking phase depends on the number of objects being tracked.

1. Kalman Filter Prediction

The predicted state $\hat{X}_{k|k-1}$ is

$$\hat{X}_{k|k-1} = F_k \hat{X}_{k-1|k-1}. \tag{16}$$

The predicted estimate covariance $P_{k|k-1}$ is

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k. \tag{17}$$

2. Kalman Filter Update

The measurement residual $\tilde{y}_k$ is

$$\tilde{y}_k = Z_k - H_k \hat{X}_{k|k-1}. \tag{18}$$

The residual covariance S_k is

$$S_k = H_k P_{k|k-1} H_k^T + R_k. \tag{19}$$

The Kalman gain K_k is

$$K_k = P_{k|k-1} H_k^T S_k^{-1}. \tag{20}$$

The updated state estimate $\hat{X}_{k|k}$ is

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k \tilde{y}_k. \tag{21}$$

The updated estimate covariance $P_{k|k}$ is

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}. \tag{22}$$

4. Implementation Details

This paper implements experiments on a PC equipped with an Nvidia GeForce RTX 3090 GPU, Intel i9-10900K 3.70 GHz CPU, and 32GB RAM.

4.1. Anchor Setting

In this article, we focus on vehicle tracking, which contains the class of car and van in the KITTI tracking dataset. It is complicated for us to select reasonable bounding box priors, which are a set of hyper-parameters because of the special appearance of these vehicles. The improper prior size may cause the predicted bounding box to be far from the corresponding ground truth and then affect the tracking performance. The number of anchor priors is set to 12 for each prediction head in our training phase. Hence, the K-means approach is applied to the tracking dataset, which only includes vehicles, since there are other classes in the entire dataset, such as pedestrians, cyclists, etc. The bounding box prior is generated in the following steps:

(1) K-means clustering. Input: all of ground truth bounding boxes of vehicles in the training dataset. Output: 12 width-height pairs of bounding boxes.
(2) Initialization. Select 12 bounding boxes from the training dataset randomly as the center of the cluster.
(3) Cluster. Calculate the IoU distance from each bounding box to the center of each cluster. Then, divide these bounding boxes into the nearest cluster.
(4) Update. With the classified 12 clusters in step (3), the cluster center in each cluster is updated according to the median of all bounding boxes.
(5) Repeat step (3) and (4) until the cluster centers are no longer changing to meet the given termination condition.

In the design of anchor boxes, 12 appropriate clusters are: (23,18), (31,22), (37,30), (60,26), (53,40), (81,38), (67,56), (122,51), (107,76), (173,94), (230,150), and (333,183).

4.2. Training and Testing

In the training phase, the model is trained for a total of 80 epochs on the synthetic dataset. The multi-step learning rate is adopted and initialized to 10^{-2}, reducing the learning rate of each parameter group by the decay factor once the number of epochs reaches one of the milestones. We set the milestones in the middle and in three-quarters of the epochs, and the decay factor is set at 0.1. The input images are resized to 1088×608 and some data augmentation techniques such as rotation, scaling and color dithering are used.

In the testing phase, the IoU threshold required to qualify as detected and the object confidence threshold are set to 0.5. The IoU threshold for non-maximum suppression is set to 0.4. The deletion condition for the final unmatched tracks is 30 frames, which means that the tracks will be removed if the condition is not satisfied. Each tracked target is assigned a Kalman filter as the motion model. Therefore, the number of objects can also affect tracking speed.

5. Experiments

5.1. Dataset and Metrics

All of the training and testing data are provided by Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [16]. The KITTI computer vision benchmark contains a large number of datasets for different tasks, such as the object detection dataset consisting of 7481 training images and 7518 test images and the tracking dataset consisting of 21 training sequences and 29 test sequences. Cars, pedestrians, and cyclists are annotated with 2D bounding boxes. We use synthetic datasets that consist of training images in both detection and tracking benchmarks. Since we study multi-vehicle tracking, annotations and images containing objects such as pedestrians are removed. Then, schemes such as image augmentation are used to enlarge the dataset during training, which improves the training results. In the following, the tracking performance is evaluated on the test sequence of the object tracking benchmark. Following the submission policy of the KITTI team, we submitted our test results to them and obtained metric scores.

Due to the complexity of multi-object tracking, there are a large number of metrics to evaluate its performance. From all video sequences, we compute more than 20 metrics by using evaluation tool that official organization provides, such as the HOTA tracking metrics (HOTA, DetA, AssA, DetRe, DetPr, AssRe, AssPr, LocA) [17], the CLEARMOT metrics (MOTA, MOTP, MT, ML, Frag, etc.) [18], identity metrics (IDs, IDSW, etc.), and fragmentation (Frag) [19] metrics. HOTA is able to comprehensively evaluate the performance of detection and data association in the TBD paradigm. While Frag focuses more on association performance, MOTA evaluates detector capabilities and focuses more on detection performance.

5.2. Experimental Results

Our tracking performance is compared with the advanced trackers (JRMOT [20], MOTSFusion [21], SRK_ODSEA [22], Quasi-Dense [23], FANTrack [24], extraCK [25], and Point3DT [26]) in recent years on the KITTI 2D car tracking benchmark. The results of these evaluation metrics, which are provided by the KITTI official team [17], are shown in Tables 1–3. The up arrow after each metric, which is in the first row of these tables, means the bigger the better, and vice versa. The HOTA metrics of the proposed method in detail are demonstrated in Figure 3. Based on the results, our algorithm achieves state-of-the-art performance. For instance, the HOTA, AssA, AssRe, and IDSW of our tracker rank first in these comparisons. Other metric scores also rank high on the list.

Table 1. Comparison results of several algorithms for eight metrics such as HOTA.

MOT-Approach	HOTA↑	DetA↑	AssA↑	DetRe↑	DetPr↑	AssRe↑	AssPr↑	LocA↑
MVT	71.00%	66.91%	75.85%	71.13%	80.72%	78.93%	85.90%	83.46%
JRMOT [20]	69.61%	73.05%	66.89%	76.95%	85.07%	69.18%	88.95%	86.72%
MOTSFusion [21]	68.74%	72.19%	66.16%	76.05%	84.88%	69.57%	85.49%	86.56%
SRK_ODSEA [22]	68.51%	75.40%	63.08%	78.89%	86.00%	65.89%	87.47%	86.88%
Quasi-Dense [23]	68.45%	72.44%	65.49%	76.01%	85.37%	68.28%	88.53%	86.50%
FANTrack [24]	60.85%	64.36%	58.69%	69.17%	80.82%	60.78%	88.94%	84.72%
extraCK [25]	59.76%	65.18%	55.47%	69.21%	81.69%	61.82%	75.70%	84.30%
Point3DT [26]	57.20%	55.71%	59.15%	64.66%	68.67%	63.20%	78.30%	80.07%

Table 2. Comparison results of several algorithms for seven metrics such as TP.

MOT-Approach	TP↑	FP↓	FN↓	MT Rate↑	PT Rate↓	ML Rate↓	FRAG↓
MVT	29,656	4736	650	66.92%	24.00%	9.08%	521
JRMOT [20]	30,325	4067	787	70.92%	24.46%	4.62%	273
MOTSFusion [21]	30,100	4292	713	72.77%	24.31%	2.92%	569
SRK_ODSEA [22]	31,062	3330	489	78.00%	19.54%	2.46%	531
Quasi-Dense [23]	30,072	4320	549	69.54%	26.61%	3.85%	567
FANTrack [24]	28,130	6262	1305	62.77%	28.46%	8.77%	701
extraCK [25]	28,463	5929	675	62.31%	31.85%	5.85%	750
Point3DT [26]	27,955	6437	4424	60.46%	26.77%	12.77%	756

Table 3. Comparison results of several algorithms for seven metrics such as MOTA.

MOT-Approach	MOTA↑	MOTP↑	MODA↑	IDSW↓	sMOTA↑	#Dets	#Tracks
MVT	84.08%	80.71%	84.34%	189	67.45%	30,306	968
JRMOT [20]	85.10%	85.28%	85.89%	271	72.11%	31,112	960
MOTSFusion [21]	84.24%	85.03%	85.45%	415	71.14%	30,813	929
SRK_ODSEA [22]	87.79%	85.41%	88.90%	380	74.62%	31,551	1039
Quasi-Dense [23]	84.93%	84.85%	85.84%	313	71.69%	30,621	979
FANTrack [24]	75.84%	82.46%	78.00%	743	61.49%	29,435	1582
extraCK [25]	79.29%	82.06%	80.80%	520	64.44%	29,138	871
Point3DT [26]	67.56%	76.83%	68.42%	294	48.73%	32,379	1086

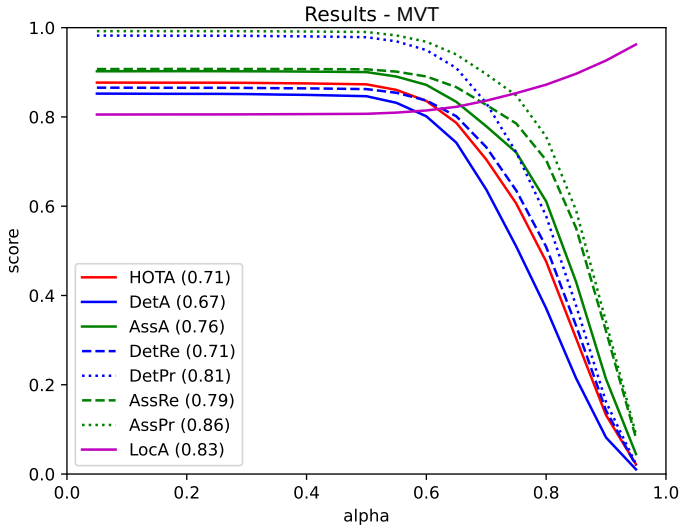

Figure 3. The detailed information of the evaluation metrics, HOTA, DetA, AssA, DetRe, DetPr, AssRe, AssPr, and LocA. The alpha is a threshold to compute the scores.

The qualitative results of our visual-based multi-vehicle tracking algorithm on KITTI [16] are shown in Figure 4. The color of each bounding box indicates the target identity. The number of frames, fps and tracked objects are shown in the upper left corner of each image. Our tracker achieves about 16 frames per second (fps) in this testing video sequence. If we optimize the structure, it can be deployed on an edge AI device that is applied to advanced driver assistance systems. In frame 52, there are six cars detected, and there are five cars detected in frame 55. Compared with the two frames, we can see that the cars (with ID 54, 55, 57, and 59) are tracked successfully. It is worth noting that the vehicle with ID 52 and 61 is not detected in frame 55 due to insignificant features and thus is not tracked. However, they are on an unstructured road that does not influence driving. The car with ID 63 appears in the field of view in frame 55 and is detected in time and given a new ID. The qualitative analysis shows that our algorithm balances real-time performance and accuracy while solving illumination changes and deformations in the image sequence.

Figure 4. Qualitative results of our visual-based tracker on KITTI. The color of each bounding box indicates the target identity. The number of frames, fps and tracked objects is shown in the upper left corner of each image.

5.3. Analysis and Discussions

To discuss the generalization of the algorithm, we analyze the algorithm visually on the test sequence 00. As shown in Figure 5, the vehicles with IDs 83, 115, 102, 113, 117, and 87 are all well detected and tracked. Although the detection of the vehicle with ID 101 between frames 113 and 117 failed at frame 175, the vehicle with ID 101 with a dark purple bounding box was re-detected in the subsequent 179 frames and retained its original ID. This means that our tracker is extremely robust.

Figure 5. Three-frame tracking results on test sequence 00 of the KITTI tracking benchmark.

In the TBD paradigm, the detection result determines the performance upper bound of the algorithm, and the data association determines whether the upper bound can be reached. To some extent, the data association capability of the proposed method achieves its detection performance. This can be seen from the comparison results in several tables, where the performance of our tracker for data association is relatively high, while the detection results are good. The performance of the vehicle-tracking algorithm is also confirmed in the qualitative analysis. Even if a target is lost in an intermediate frame, subsequent targets can still be assigned the same ID. This is also due to the excellent performance of the motion model and the matching mechanism that incorporates embedding.

On the KITTI dataset, the frame rate of our algorithm is about 16 fps, which is determined by both the size of the input image and the number of objects.

In the future, we will consider three technologies to optimize the runtime of the deployment on Nvidia devices. First, the architecture of the detection model can be changed according to its size, depth, and width. Secondly, quantization can be used to change the digital representation of data and network weights. For example, we can replace floating-point numbers with integers. Finally, we will choose the appropriate image resolution. We will conduct hardware acceleration and quantitative experiments with TensorRT on the target platform. For TensorRT reasoning, we can export the trained PyTorch model to ONNX and parse it into the optimized TensorRT runtime engine in a C++ environment on the target system. TensorRT allows us to choose the required quantization when building the engine. There are three options: Float32, Float16 and Int8.

6. Conclusions

This article proposes a multi-vehicle tracking method suitable for advanced driver assistance systems based on monocular cameras in the driver view. The approach is based on the tracking-by-detection paradigm. At the same time, joint detection and appearance embedding are used to improve tracking speed. In the detection phase, we use a modified one-stage detection network equipped with an adjusted BiFPN block with three prediction heads consisting of a sequence of convolutional neural networks. In addition, the k-means algorithm is used on the training dataset to obtain proper priors. On the other hand, the tracking phase mainly consists of a two-step matching method and a Kalman filter. Finally, experimental results demonstrate that our approach achieves state-of-the-art performance on public benchmarks and that it can handle deformation and illumination variations. While maintaining high performance, our tracker can achieve a speed of about 16 FPS on the KITTI tracking dataset. Raw data and demos are available at https://github.com/LvpengfeiNJ/MVT-MCDV-ADAS-Tracking (accessed on 24 October 2022).

In the future work, we will further explore the number of convolution layers of the prediction head which can make the detection performance the best. We will also study how to deploy this multi-vehicle tracking algorithms to edge AI devices and look into ways to minimize the loss of performance during deployment.

Author Contributions: Conceptualization, P.L.; Investigation, Y.W.; Methodology, P.L.; Project administration, M.W.; Resources, M.W.; Software, P.L.; Supervision, M.W.; Validation, P.L.; Visualization, Y.W.; Writing—original draft, P.L.; Writing—review and editing, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to my multi-vehicle tracking enthusiasts for their suggestions on my program. Thanks to the KITTI team for evaluating my submission of the tracking results.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TBD	Tracking-by-detection
DA	Data association
MVT	Multi-vehicle tracking
JDE	Joint detection and embedding
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute

References

1. Tang, Z.R.; Hu, R.; Chen, Y.; Sun, Z.H.; Li, M. Multi-expert learning for fusion of pedestrian detection bounding box. *Knowl. Based Syst.* **2020**, *241*, 108254. [CrossRef]
2. Kumar, A.; Saini, T.; Pandey, P.B.; Agarwal, A.; Agrawal, A.; Agarwal, B. Vision-based outdoor navigation of self-driving car using lane detection. *Int. J. Inf. Technol.* **2022**, *14*, 215–227. [CrossRef]
3. Meinhardt, T.; Kirillov, A.; Leal-Taixe, L.; Feichtenhofer, C. Trackformer: Multi-object tracking with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 8844–8854.
4. Zhang, Y.; Sun, P.; Jiang, Y.; Yu, D.; Weng, F.; Yuan, Z.; Wang, X. Bytetrack: Multi-object tracking by associating every detection box. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 1–21.
5. Ciaparrone, G.; Sánchez, F.L.; Tabik, S.; Troiano, L.; Tagliaferri, R.; Herrera, F. Deep learning in video multi-object tracking: A survey. *Neurocomputing* **2020**, *381*, 61–88. [CrossRef]
6. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
7. Wang, Z.; Zheng, L.; Liu, Y.; Li, Y.; Wang, S. Towards real-time multi-object tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 107–122.
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *1*, 91–99. [CrossRef] [PubMed]
9. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
10. Auger, F.; Hilairet, M.; Guerrero, J.M.; Monmasson, E.; Orlowska-Kowalska, T.; Katsura, S. Industrial applications of the Kalman filter: A review. *IEEE Trans. Ind. Electron.* **2013**, *60*, 5458–5471. [CrossRef]
11. Hamuda, E.; Mc Ginley, B.; Glavin, M.; Jones, E. Improved image processing-based crop detection using Kalman filtering and the Hungarian algorithm. *Comput. Electron. Agric.* **2018**, *148*, 37–44. [CrossRef]
12. Bergmann, P.; Meinhardt, T.; Leal-Taixe, L. Tracking without bells and whistles. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 941–951.
13. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10781–10790.
14. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. Baisa, N.L. Derivation of a Constant Velocity Motion Model for Visual Tracking. *arXiv* **2020**, arXiv:2005.00844.
16. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
17. Luiten, J.; Osep, A.; Dendorfer, P.; Torr, P.; Geiger, A.; Leal-Taixé, L.; Leibe, B. Hota: A higher order metric for evaluating multi-object tracking. *Int. J. Comput. Vis.* **2021**, *129*, 548–578. [CrossRef] [PubMed]
18. Bernardin, K.; Stiefelhagen, R. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP J. Image Video Process.* **2008**, *2008*, 1–10. [CrossRef]
19. Li, Y.; Huang, C.; Nevatia, R. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, USA, 20–25 June 2009; pp. 2953–2960.
20. Shenoi, A.; Patel, M.; Gwak, J.; Goebel, P.; Sadeghian, A.; Rezatofighi, H.; Savarese, S. Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 10335–10342.
21. Luiten, J.; Fischer, T.; Leibe, B. Track to reconstruct and reconstruct to track. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1803–1810. [CrossRef]
22. Mykheievskyi, D.; Borysenko, D.; Porokhonskyy, V. Learning local feature descriptors for multiple object tracking. In Proceedings of the Asian Conference on Computer Vision, Kyoto, Japan, 30 November–4 December 2020.
23. Pang, J.; Qiu, L.; Li, X.; Chen, H.; Li, Q.; Darrell, T.; Yu, F. Quasi-dense similarity learning for multiple object tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 164–173.
24. Baser, E.; Balasubramanian, V.; Bhattacharyya, P.; Czarnecki, K. Fantrack: 3d multi-object tracking with feature association network. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 1426–1433.
25. Gündüz, G.; Acarman, T. A lightweight online multiple object vehicle tracking method. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 427–432.
26. Wang, S.; Sun, Y.; Liu, C.; Liu, M. Pointtracknet: An end-to-end network for 3-d object detection and tracking from point clouds. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3206–3212. [CrossRef]