electronics

Article

Multi-Camera Vehicle Tracking Based on Deep Tracklet Similarity Network

Yun-Lun Li, Hao-Ting Li and Chen-Kuo Chiang *

Advanced Institute of Manufacturing with High-Tech Innovations, Center for Innovative Research on Aging Society (CIRAS) and Department of Computer Science and Information Engineering, National Chung Cheng University, Minhsiung, Chiayi 621301, Taiwan; xu3mp6xjp6@gmail.com (Y.-L.L.); remidream@gmail.com (H.-T.L.)
* Correspondence: ckchiang@cs.ccu.edu.tw; Tel.: +886-5-272-9111

Abstract: Multi-camera vehicle tracking at the city scale has received lots of attention in the last few years. It has large-scale differences, frequent occlusion, and appearance differences caused by the viewing angle differences, which is quite challenging. In this research, we propose the Tracklet Similarity Network (TSN) for a multi-target multi-camera (MTMC) vehicle tracking system based on the evaluation of the similarity between vehicle tracklets. In addition, a novel component, Candidates Intersection Ratio (CIR), is proposed to refine the similarity. It provides an associate scheme to build the multi-camera tracking results as a tree structure. Based on these components, an end-to-end vehicle tracking system is proposed. The experimental results demonstrate that an 11% improvement on the evaluation score is obtained compared to the conventional similarity baseline.

Keywords: vehicle tracking; multiple camera; tracklet similarity; deep learning

Citation: Li, Y.-L.; Li, H.-T.; Chiang, C.-K. Multi-Camera Vehicle Tracking Based on Deep Tracklet Similarity Network. *Electronics* **2022**, *11*, 1008. https://doi.org/10.3390/electronics11071008

Academic Editor: John Ball, Ning Wang

Received: 7 December 2021
Accepted: 22 March 2022
Published: 24 March 2022

1. Introduction

With the recent advancement of computer vision, city-scale automatic traffic management is now possible. Real-time multi-target multi-camera (MTMC) vehicle tracking can be improved by techniques for automatic traffic monitoring and management [1–6]. Automatic video analytics can enhance traffic infrastructure design and congestion handling through the pervasively deployed traffic cameras.

Real-time multi-target multi-camera tracking is one of the crucial tasks in traffic management. Its purpose is to achieve better traffic design and traffic flow optimization by tracking many vehicles in a network across multiple surveillance cameras, as shown in Figure 1. Most approaches in MTMC follow the tracking by detection pipeline. Firstly, a detector is adopted to obtain all vehicle detections. After vehicle detection, a single-camera tracker needs to form vehicle tracklets of the same vehicle in each view. Then, these vehicle tracklets are associated across cameras.

There are also several difficulties for the MTMC task. The problems of how to eliminate unreliable vehicle tracklets and deal with view variations are significant in these tasks. Large-scale automatic video analytic systems must handle a large variability of vehicle types and appearances to meet the accuracy and reliability requirements in the real world. For applications such as vehicle re-identification, large view variations cast a significant challenge in vehicle re-identification across views. Similarly, how best to perform space–time vehicle tracklet association across views is important for vehicle counting and traffic analysis. In addition, images are captured by different cameras. The vehicle may have different poses and illumination conditions, resulting in different colors of the appearances. Different weather conditions, such as raining or hazing, make vehicle tracking problems more challenging.

Existing works [7,8] evaluate the connectivity between tracklets across cameras by simple Euclidean distance and cosine similarity. However, these metrics are not robust enough to measure the connectivity in tracklets. Moreover, when one tracklet is associated

only with a tracklet having the highest similarity score, it does not fully use the information of the whole ranking list. To address these problems, a novel Tracklet Similarity Network (TSN) and Candidate Intersection Ratio (CIR) metric is proposed. TSN can better learn robust similarity scores. CIR and its corresponding association scheme can better leverage the ranking information of the tracklets.

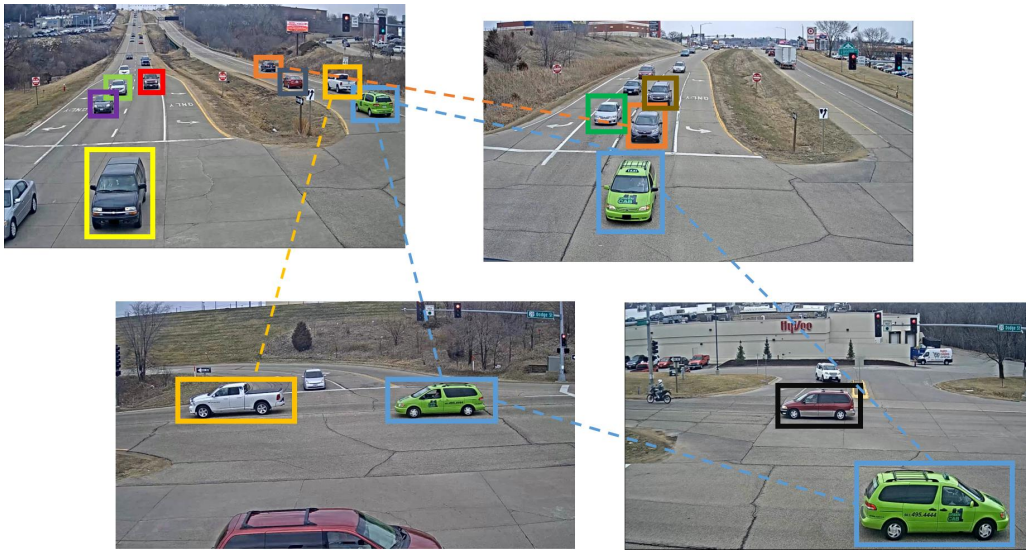

Figure 1. A multi-target multi-camera tracking example. The four subfigures show different views captured by cameras. The bounding box with the same color refers to the same car across cameras.

In this paper, the proposed MTMC system is proposed via a complete pipeline. Firstly, Faster-RCNN is adopted as the vehicle detection module. Then, the TrackletNet tracker can generate tracklets based on vehicle detection results. Filters are applied to eliminate unreasonable tracklets. Finally, the proposed TSN evaluates similarities between tracklets, and the CIR matching scheme associates tracklets across cameras. To sum up, the contributions of this paper include the following. (1) Several novel components are proposed in this paper. Re-Id Block learns the feature representation for each tracklet rather than applying CNN to extract image features. Disentangle Block separates the tracklet feature from cameras to learn a camera-independent feature by exploiting camera loss and the ranked list loss functions. Similarity Measurement Block leverages strong similarities information from the ranking list of tracklets rather than using the tracklet with the highest score alone via a random walk scheme. (2) The aforementioned components are exploited to build an end-to-end deep multi-camera vehicle tracking model. (3) The experimental results demonstrate that an 11% improvement on the evaluation score is obtained compared to the conventional similarity baseline. The proposed CIR outperforms the pairing method over 3.6% by the Cosine Similarity and nearly 2% by the IDF1 score.

2. Related Work

2.1. Single-Camera Tracking

The tracking-by-detection paradigm has been dominant in single-camera tracking tasks. DeepSORT [9] combines the Kalman filter [10] and Hungarian algorithm [11]. The Kalman filter helps predict the next position of a tracklet. The Hungarian algorithm is a data association method. It is used to associate tracklets. However, tracking-by-detection is a time-consuming process. The JDE tracker [12] is proposed to address this problem. The JDE tracker can generate vehicle detection and embeddings simultaneously so that it can save significant time. Yang et al. [2] propose a tracklet reconnection technique in SCT. They leverage GPS information and pre-defined zone areas to refine tracklet results.

2.2. Multi-Camera Tracking

Multi-camera tracking aims to associate tracklets across cameras. Most approaches follow the pipeline of vehicle detection, vehicle feature extraction, single-camera tracking, and multi-camera tracking. These approaches can be classified into three categories: vehicle feature extraction, hierarchical structure, and exception handling methods. Many multi-camera tracking methods focus on representative feature extraction for tracking. He et al. [1] propose a spatial–temporal attention mechanism in their feature extraction module. Therefore, they can generate more robust tracklet features. Ristani et al. [13] proposed adaptive weighted triplet loss for learning appearance features and hard identity mining for sampling hard negatives more frequently. Peri et al. [14] introduced an unsupervised excitation layer for highlighting the discriminative saliency region. The saliency map can increase the robustness and performance of the model. However, these methods fail to consider feature disentanglement and the variations brought by different cameras.

The second category is hierarchical structure methods. Li et al. [3] develop a hierarchical match strategy by the rule-based method. When associating tracklets, many rules can be considered, such as vehicle speed, directions, motion, etc. Xu et al. [5] presented a hierarchical composition model to composite the geometry, appearance, and motion of trajectories for multi-view multi-object tracking. Xu et al. [15] exploited semantic attributes by building a parse tree to relate the same people on different camera views for multi-camera people tracking. Wen et al. [16] achieve multi-camera multi-target tracking by formulating the problem as a sampling-based search of the space–time view hypergraph. The hypergraph is able to encode higher-order constraints on 3D geometry, motion continuity, and trajectory among 2D tracklets within and across different camera views. These methods capture multi-view objects by adding the physical properties of objects. However, their formulations usually have many parameters for different terms. The parameter tuning is difficult and time consuming.

The third category is exception handling methods. You et al. [17] utilizes local pose matching to solve the occlusion problem for multi-target multi-camera tracking. Optical flow is applied to reduce the distance caused by fast motion. The idea can be extended from pedestrians to vehicles easily. Specker et al. [18] consider that most tracking errors of false detection occur in occlusions. They develop an occlusion handling approach by labeling the overlap higher than a non-maximum suppression threshold as an occlusion pair. The occlusion information is leveraged for removing false detections and multi-camera clustering for tracking.

Bredereck et al. [6] proposed a data association method with the greedy algorithm to associate trackers for multi-object tracking. Some physical constraints are used to make the tracklet reliable. The aforementioned research evaluates the similarity between tracklets by simple cosine similarity, Euclidean distance, or handcrafted metric. Instead of using evaluation metrices, Hou et al. [4] proposed a network to measure the similarity. They adopt temporal sliding windows in their data sample method to capture the local temporal feature of tracklets. In comparison, our method proposes deep neural network learning features across cameras and exploits the Random Walk method to refine the similarity by further considering the relationships between tracklets in the target gallery.

2.3. Random Walk

The Random Walk algorithm is a graphical method to determine the probability of one point to another point. An initial probability can be defined for the graph that decides which point to start. For each walking step, the point-to-point probability is updated. After a few iterations, the probability will be stabilized. Ref. [19] proposed a random walk layer in deep neural networks. The network calculates the refined affinities between probe image and gallery images. By calculating cross-entropy loss with refined affinities, the network can be updated with stronger supervision. Motivated by [19], we exploit and design an additional random walk layer in our model.

2.4. Ranked List Loss

Triplet Loss [20] is widely used for learning similarity and dissimilarity semantic information of data. The Triplet Loss is computed to ensure that an anchor image x_i^a is closer to positive samples x_i^p while pushing away from negative samples x_i^n and maintaining a margin distance m in feature space f:

$$\mathcal{L}_{triplet} = \sum_i \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + m \right]_+ \quad (1)$$

where $[\cdot]_+$ denotes the hinge function. However, it is known to cause slow convergence. Ranked List Loss (RLL) [21] is proposed to address this problem by pushing negative samples from a boundary α, maintaining a margin m between positive samples and negative samples, and pulling closer positive samples within a boundary $\alpha - m$. This paper adopts Ranked List Loss for features disentanglement to maintain intra-class compactness and inter-class separation; this method helps learn more discriminative features.

3. Proposed Method

Following the tracking-by-detection paradigm, the framework of vehicle tracking is composed of three stages: (1) vehicle detection, (2) single-camera tracking, and (3) multi-camera tracking. In the first stage, video frames are extracted and used as the input into a vehicle detector. The bounding boxes are determined and passed into a feature extraction module. In the single-camera tracking stage, the bounding boxes under the same camera are associated altogether to build vehicle tracklets. Parked cars or bounding boxes with no car are likely to appear in the vehicle tracklets. Therefore, three filters are designed to remove false positive samples. In the last stage, the Tracklet Similarity Network is used to calculate the similarity between tracklets. The Candidates Intersection Ratio is used to refine the similarity and obtain the final similarity results. A CIR association scheme can represent the same tracklet under different cameras as a tree structure. The pipeline of the system framework is depicted in Figure 2.

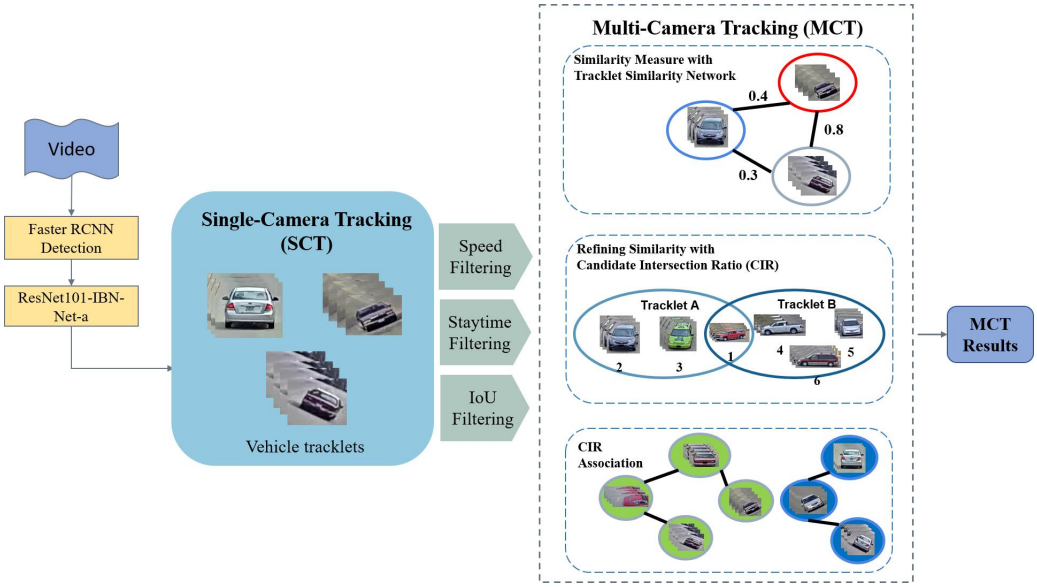

Figure 2. Multi-target multi-camera tracking pipeline.

3.1. Revisiting of Random Walk

The Random Walk algorithm is widely used to refine the probability to a stable status. Let $A \in \mathbb{R}^{n \times n}$ denote the pairwise affinities of gallery objects, where n is the number of gallery objects, and f_0 denote the initial affinities between the query object and other objects.

To keep the sum of each row equal to one $\sum_j A(i,j) = 1$, let S store the value of A. $A(i,j)$ can be normalized as follows.

$$A(i,j) = \frac{\exp S(i,j)}{\sum_{i \neq j} \exp S(i,j)} \tag{2}$$

$A(i,j)$ is set to 0 for $i = j$. Then, the distribution can be updated by Equation (3).

$$f_t = f_0 A^t \tag{3}$$

where f_t denotes the distribution after t iterations.

To avoid the affinities deviating too far away from the initial affinities, the refined affinities and initial affinities are combined.

$$f_t = \lambda A f_{t-1} + (1-\lambda) f_0 \tag{4}$$

After several derivation steps, the equation can be re-written as follows.

$$f_\infty = (1-\lambda)(I - \lambda A)^{-1} f_0 \tag{5}$$

In this paper, we adopt Equation (5) to integrate the Random Walk algorithm in the proposed deep network.

3.2. Tracklet Similarity Network

The Tracklet Similarity Network (TSN) is proposed to calculate the similarity between tracklets. TSN consists of three building blocks, including ReID Block, Disentangle Block, and Similarity Measurement Block, as depicted in Figure 3.

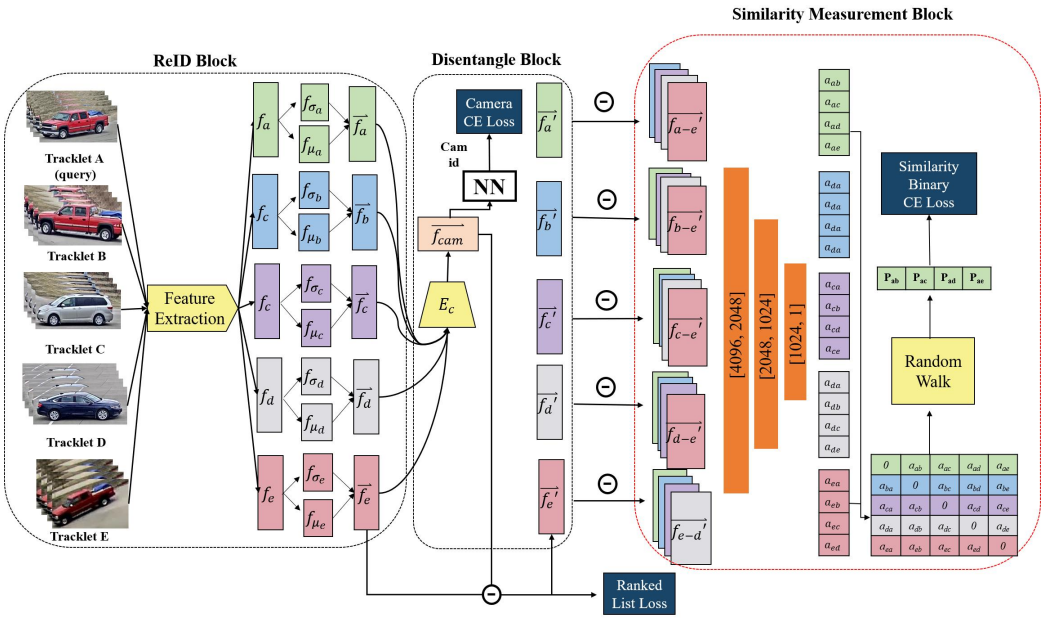

Figure 3. The architecture of the Tracklet Similarity Network.

ReID Block. The ReID block aims to learn the feature representation of the tracklet. The ResNet series includes the most popular CNN models in the computer vision field. IBN-Net is usually used to enhance ResNet, since it is appearance-invariant without additional computational consumption. Therefore, ResNet-101-IBN-Net becomes a common backbone model for feature extraction in the vehicle re-identification problem. ResNet101-IBN-Net-a [22] is exploited to extract the feature for each video frame, which is denoted as $\tilde{f}$. The mean and standard deviation of features extracted from each frame within the tracklet are

calculated and concatenated. This forms the tracklet feature by the 4096-dimension. Denote $\vec{f}$ as the tracklet feature. The feature extraction module is fixed during the training process.

Disentangle Block. Videos from different cameras may have different color tones or brightness. Basically, camera specification has its impact on the image feature learned from images from that camera. Therefore, using the camera ID as the answer of classification enforces the deep model to learn camera-specific features. The is the reason to introduce Camera Cross-Entropy Loss to learn the classifier and extract the camera feature. To reduce the impact brought by cameras, we can disentangle the camera feature from the tracklet feature. Technically, the following process can be applied.

The feature representation learned from the ReID Block is used as the input into the camera encoder to learn the camera feature, $\vec{f_{cam}}$. Several fully connected layers are served as the classifier for camera ID classification. The learning of camera features is optimized via the cross-entropy loss of camera ID. Next, the camera feature is subtracted from the tracklet feature; the feature without camera information denotes $\vec{f}'$. Finally, we use the ranked list loss to ensure that the learned disentangled feature can distinguish the difference between tracklets. The details of ranked list loss are introduced in the next subsection.

Similarity Measurement Block. The similarity measurement block leverages the disentangled feature to measure the similarity between each tracklet. Denote $\vec{f_{a-e}}$ the residual by subtracting $\vec{f}'_e$ from $\vec{f}'_a$. This extracts more discriminant features for each tracklet. Three fully connected layers are exploited to measure the similarity between each tracklet. Each dimension of the output represents the similarity score between one tracklet to another tracklet. Finally, a random walk is used to refine the similarity. The network is optimized with the refined similarity by cross-entropy loss. Although the random walk matrix provides full affinities for each pair of the tracklets, only the similarity scores between the query and the gallery are required. One can note that during the calculation of similarity between the query and the gallery, the similarities between gallery tracklets are considered altogether. Once we consider the similarity between the query tracklet A to the gallery tracklet B, the similarity is affected by the similarities between tracklet B and tracklet C, D, and E as well. This leverages the information not only between query and gallery but also similarities between gallery tracklets.

Loss Function. The cross-entropy loss is used when learning the camera features for each tracklet. The *camera cross-entropy loss* can be formulated as follows.

$$\mathcal{L}_{CAM} = -\frac{1}{N}\sum_{i=1}^{N} y^i_{cam} \log(\hat{y}^i_{cam}) \tag{6}$$

where $\hat{y}^i_{cam}$ is the camera ID of the prediction, and y^i_{cam} is the target.

The binary classifier distinguishes tracklet similarity through refined affinities from the random walk. Similarly, the classifier is optimized by the *similarity binary cross-entropy loss* using the following formula:

$$\mathcal{L}_{SIM} = -\frac{1}{N}\sum_{i=1}^{N} y^i_{sim} \log(\hat{y}^i_{sim}) + (1 - y^i_{sim}) \log(1 - \hat{y}^i_{sim}) \tag{7}$$

where $\hat{y}^i_{sim}$ is the similarity of the prediction, and y^i_{sim} is the target.

The ranked list loss is adopted to distinguish positive and negative samples. The ranked list loss is defined as:

$$\mathcal{L}_m(x_i, x_j) = (1 - y_{ij})[\alpha - d_{ij}]_+ + y_{ij}[d_{ij} - (\alpha - m)]_+ \tag{8}$$

where $[\cdot]_+$ refers to the hinge function, and

$$\mathcal{L}_{RLL} = \frac{1}{N}\sum_{\forall c,\forall i}\left(0.5\sum_{x_j^c\in|P_{c,i}^*|}\frac{1}{P_c}\mathcal{L}_m(x_i^c,x_j^c)\right)+\left(0.5\sum_{x_j^c\in|N_{c,i}^*|}\frac{1}{N_c}\mathcal{L}_m(x_i^c,x_j^c)\right) \tag{9}$$

where x denotes the tracklet feature, and $y_{ij}\in\{0,1\}$ denotes whether the identity of x_i and x_j is the same or not. If x_i and x_j are the same identity, y_{ij} is 1. Otherwise, it is 0. d_{ij} denotes the Euclidean distance between x_i and x_j. α is the negative boundary, and m represents the margin of the positive and the negative sample. To minimize $\mathcal{L}_m$, the distance between the data sample and the positive sample is constrained to be lower than $\alpha - m$, while the distance between the data sample and the negative sample aims to be larger α. In the loss function $\mathcal{L}_{RLL}$, x_i^c means data sample x_i is from class c. $|P_{c,i}^*|$ denotes the set of all positive samples to x_i^c. Thus, the first term of Equation (9) calculates the average distance of data sample x_i and all its positive samples over all data samples and all classes. Similarly, $|N_{c,i}^*|$ denotes the set of all negative samples to x_i^c. $\mathcal{L}_{RLL}$ calculates the average distance from data sample to its positive negative samples.

The total loss for the Tracklet Similarity Network is defined as follows.

$$\mathcal{L}_{total} = \lambda_{RLL}\mathcal{L}_{RLL} + \lambda_{CAM}\mathcal{L}_{CAM} + \lambda_{SIM}\mathcal{L}_{SIM} \tag{10}$$

3.3. Candidates Intersection Ratio

We develop the CIR tracklet matching metric to evaluate the similarity between two tracklets for the calculation and ranking of tracklet associations iteratively, and the association will be performed hierarchically similar to the standard **agglomerative clustering** algorithm in *hierarchical clustering*. We follow a similar notation in considering a tracklet A with matching candidates L_A and another tracklet B with matching candidates L_B. The CIR metric for evaluating the association similarity of the matching list of tracklets L_A and L_B is defined as:

$$CIR(L_A, L_B) = \frac{\text{size}(L_A \cap L_B)}{\min(\text{size}(L_A), \text{size}(L_B))} \tag{11}$$

Figure 4 shows an example of the CIR metric calculation, where the intersection of the matching list of two tracklets is calculated according to Equation (11) to determine if the two tracklets should be associated together (i.e., merged or linked, and regarded as the tracklets of the same vehicle across views).

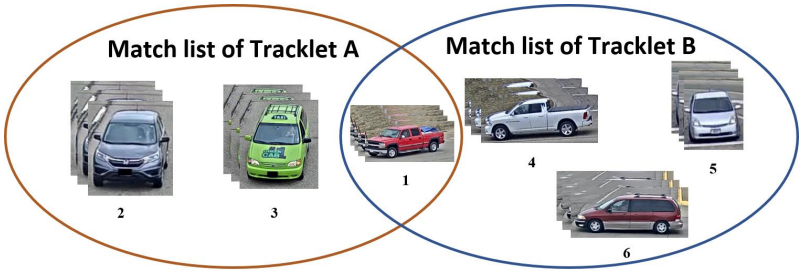

Figure 4. A Candidates Intersection Ratio example.

The **CIR tracklet association algorithm** performs iteratively by considering all such (L_A, L_B) pairs across views. Initially, all distinct (unassociated) tracklets are regarded as individual nodes without edges connecting them. The matching list of associated tracklets is constructed by associating the pair (L_A, L_B), and repeated association yields a binary tree-like structure, where the root of the tree indicates the vehicle ID of this association tree. This way, the association tree is constructed by regarding tracklets and their CIR similarity as nodes and edge weight. During tree construction, we enforce that the size of the parent node must be larger or equal to the size of the children nodes. This way, a unique vehicle ID should be retained for each tree throughout the whole iterative association process.

We iteratively compute the CIR score between every tracklet and the tracklets of sizes smaller or equal to the first one. Figure 5 illustrate an example of the tracklet association. In the association of tracklet A to its candidates, we find that tracklet B contains a larger CIR score with tracklet A (where the score $>$ threshold). Therefore, tracklet A is assigned as the parent node of tracklet B. By repeating this process, we can cluster the same tracklets as a tree.

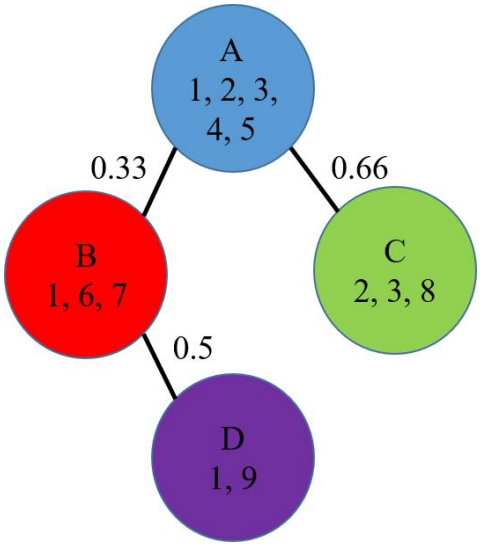

Figure 5. A tracklet association example during the execution of the CIR association. Suppose tracklet B has the max CIR with tracklet A. Since the size of tracklet B is smaller than the size of tracklet A, tracklet A is assigned as tracklet B's parent. We repeat this process and group the tracklets as a tree.

In contrast to a naive greedy best-first tracklet association algorithm that only considers the association of the best-matching pair of tracklets, the proposed CIR tracklet association considers the matching of the whole matching list of tracklets for optimization.

Avoid cyclic tracklet associations. Repeated tracklet association following the above steps by associating tracklets with the best-first selection of CIR scores might result in an unwanted case of *cycle associations*, where the resulting vehicle ID of the associated set cannot be uniquely defined, causing problematic MTMC results. To avoid such an issue, we enforce the following rules during the CIR association steps to explicitly check and avoid cyclic associations.

1. When the size of the matching list of tracklet A is larger than the size of the matching list of tracklet B, we assign tracklet B as the child node of tracklet A.
2. If the size of the matching list of tracklet A is equal to the size of the matching list of tracklet B, we check and make sure that tracklet B is not the ancestor of A. If tracklet B is indeed the ancestor of A, we assign tracklet B to be the parent of tracklet A. Otherwise, we assign tracklet A to be the parent of tracklet B.

The CIR association iteration terminates when all pairwise tracklet lists are examined. Afterwards, we assign the vehicle ID of each tree root to all of its descent nodes. This completes the CIR multi-camera vehicle tracking. Algorithm 1 shows the detailed steps of the CIR vehicle tracklet association algorithm in pseudo code.

Algorithm 1: CIR Tracklet Association.

Input: tracklets $T_1, ..., T_N$, number of tracklets N
Output: tracklets after association $T'_1, ..., T'_N$
for $i = 1$ *to* N **do**
  Calculate the pair-wise cosine similarity $S_{i,j}$
  Obtain $T_i.L$ the matching list of T_i using $S_{i,j}$
  $T_i.L := L$
  //initialize CIR threshold
  $T_i.max\_score := 0.33$
for $i = 1$ *to* N **do**
  for $j = 1$ *to* N **do**
    if $(i == j)$ *or* $(T_i.camera == T_j.camera)$ *or* $(size(T_i) < size(T_j))$ **then**
      continue
    $score := CIR(T_i.L, T_j.L)$
    if $size(T_i) == size(T_j)$ **then**
      if $is\_ancestor(T_i, T_j)$ **then**
        if $score > T_i.max\_score$ **then**
          $T_i.parent := T_j$
          $T_i.max\_score := score$
      else
        if $score > T_j.max\_score$ **then**
          $T_j.parent := T_i$
          $T_j.max\_score := score$
    else
      if $score > T_j.max\_score$ **then**
        $T_j.parent := T_i$
        $T_j.max\_score := score$
for $i = 1$ *to* N **do**
  $T'_i.id := T_i.root\_id$

3.4. Other System Components

Vehicle Detection and Single-Camera Tracking. In the begining of the multi-target multi-camera tracking pipeline of Figure 2, Faster R-CNN [23] is adopted as the detector using ResNet-101 as the backbone model. TrackletNet Tracker [24] is used for single-camera tracking. TrackletNet Tracker is a graphical model that effectively reduces computational complexity with temporal information and predicts bounding boxes with epipolar geometry.

Single-Camera Tracklets Filtering. As shown in Figure 6, there is an empty bounding box. The empty bounding box occurs in two situations: (1) the car is at high speed, and (2) the car stays for a long time. We propose the tracklet speed filter and the tracklet stay time filter to deal with the two situations. In addition, we propose the tracklet IoU filter to eliminate the parked car.

Figure 6. Unreliable tracklet examples.

The tracklet speed filter is defined as:

$$\frac{v - \mu_v}{\sigma_v} > \tau_v \tag{12}$$

where v is the tracklet speed, μ_v is the average speed in the same camera, σ_v is the standard deviation of the speed, and τ_v is the threshold. The tracklet speed is computed with the GPS position and duration time. The GPS position and duration time are projected from the calibration homography matrix provided by AICITY Organization. In this way, the tracklet speed filter removes the outlier of the tracklet speed. The number of removed tracklets is 3% manually controlled. Similarly, the tracklet stay time filter is defined as:

$$\left| \frac{t - \mu_t}{\sigma_t} \right| > \tau_t \tag{13}$$

where t is the stay time of the tracklet, μ_t is the average stay time in the same camera, σ_v is the standard deviation of the stay time, and τ_v is the threshold. With the two filters, we can remove most of the empty bounding boxes.

The proposed tracklet IoU filter is defined as:

$$\frac{box_s \cap box_e}{box_s \cup box_e} > \tau_{iou} \tag{14}$$

where box_s refers to the first bounding box, and box_e refers to the last bounding box. If the IoU is greater than the threshold, the car in the tracklet does not move. The filters eliminate the candidate bounding boxes, thereby reducing mismatched tracklets.

The overview pipeline is described in Algorithm 2. The Single-Camera Tracking module (SCT) uses video frames as input and outputs the raw tracklets T_1 to $T_{N_{raw}}$. By conducting three filters: speed, staytime, and IoU, tracklets can be refined as T_1 to T_N. Then, they are used as input to the Tracklet Similarity Network (TSN). The output of TSN is the top N rank list of each query tracklet, which is represented as a matrix of $S^{N \times N}$. After processing the Candidates Intersection Ratio (CIR), it outputs the final associated tracklets.

Algorithm 2: Multi-target multi-camera tracking pipeline.

Input: video frames $f_1, \ldots, f_K$
Output: tracklets after association $T'_1, ..., T'_N$
$[T_1, \ldots, T_{N_{raw}}] \coloneqq \text{SCT}(f_1, \ldots, f_K)$
$[T_1, \ldots, T_{N_{sp}}] \coloneqq \text{filter}_{speed}([T_1, \ldots, T_{N_{raw}}]))$
$[T_1, \ldots, T_{N_{st}}] \coloneqq \text{filter}_{staytime}([T_1, \ldots, T_{N_{sp}}]))$
$[T_1, \ldots, T_N] \coloneqq \text{filter}_{iou}([T_1, \ldots, T_{N_{st}}]))$
$\mathbf{S}^{N \times N} \coloneqq \text{TSN}([T_1, \ldots, T_N])$
$[T'_1, \ldots, T'_N] \coloneqq \text{CIR}_{association}([T_1, \ldots, T_N], \mathbf{S})$

4. Experimental Results

4.1. Implementation Details

The Disentangle Block comprises fully connected layers; the number of neurons is given by 4096–4096–32. After this, the Similarity Measurement Block comprises three fully connected layers of 4096–2048–1024–1, as illustrated in Figure 3. All fully connected layers in TSN except the classifier are initialized following [25]. In the classifier, the weights are sampled from the normal distribution with a mean of 0 and a standard deviation of 0.001, and the biases are set to 0.

In the single-camera filtering stage, we set the three threshold $\tau_{iou} = 0.05$, and $\tau_s = \tau_v = 1$. If the number of removed tracklets exceeds 3%, the thresholds are added to 0.5. This procedure is repeated until the number of removed tracklets is less than 3%. We adopt Stochastic Gradient Descent (SGD) as the optimizer, set the learning rate to 0.001,

and set the weight decay to 0.3. The α and m are set to 1.4 and 0.8, respectively. Following the hyperparameter settings, the training process is converged in five epochs. In the CIR association scheme, we set $\tau_{sim} = 0.8$ to determine which candidates can be selected into the match list. The hyperparameters are listed in Table 1.

The performance analysis of our method is 12.94 fps to process one image during testing and 6.6 h for training. The evaluation is performed with an AMD Ryzen 9 5950X CPU and NVIDIA GeForce RTX 3090 GPU. All experiments are repeated three times, and the average results are reported in the following sections.

Table 1. Hyperparameter settings.

Hyperparameter	Value
Hidden layers in the Disentangle Block	4096-4096-32
Hidden layers in the Similarity Measurement Block	4096-2048-1024-1
Optimizer	SGD
Learning rate	0.001
Weight decay	0.3
Batch size	16
Number of epochs	5
α	1.4
m	0.8
τ_{iou}	0.05
τ_s	1
τ_v	1
λ_{RLL}	1.0
λ_{CAM}	1.0
λ_{SIM}	1.0

4.2. Dataset

We evaluate our method on the CityFlowV2 dataset [26,27]. The dataset contains six scenes captured by a total of 46 cameras. The length of the videos is 215 min. In the dataset, three scenes are used for training, two are used for validation, and one is used for testing. The car is annotated if at least two cameras capture it.

The dataset comprises hard samples and easy samples. A hard sample is a sample that cannot be correctly sorted by cosine similarity. Otherwise, it is an easy sample. For each sample, we choose a query tracklet and the corresponding positive and negative gallery tracklets. The data sample obeys the following rules:

1. The number of positive and negative gallery tracklets must be consistent.
2. The data sample in which the similarity between any negative gallery tracklet and a query tracklet is greater than the similarity between the positive gallery tracklet and the query tracklet is a hard sample; otherwise, it is an easy sample.
3. The ratio of hard samples and easy samples is kept to 1:1.

We sample 668 training data and 668 validation data following these rules.

4.3. Evaluation Metric

*IDF*1, *IDP*, and *IPR* [28] are common evaluation indicators for multi-camera tracking and single-camera tracking and are usually the default indicators for evaluating tracker capabilities. The formula is defined as follows:

$$IDF1 = \frac{2IDTP}{2IDTP + IDFP + IDFN} \tag{15}$$

$$IDP = \frac{IDTP}{IDTP + IDFP} \tag{16}$$

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{17}$$

where $IDTP$, $IDFP$, and $IDFN$ represent the true positive, false positive, and false negative of the vehicle ID, respectively.

MOTA is one of the indicators used to evaluate a single-camera tracker. It does not calculate the accuracy of the object position but uses the missing rate, misjudgment rate, and mismatch rate to evaluate the quality of a tracker. The formula of *MOTA* is listed as follows.

$$MOTA = 1 - \sum_t \frac{(FN_t + FP_t + \phi_t)}{GT_t} \tag{18}$$

where t is the index of the number of frames, FN, FP, and GT are the number of false negatives, false positives, and ground truths. ϕ is the number of ID switches.

4.4. Ablation Study

Table 2 presents the results of single-camera tracking filtering. For the baseline method, parked cars or bounding boxes with no car are likely to appear in the vehicle tracklets and thus decrease the performance. The proposed three filters improve the single-camera tracking task by 20% in *IDF*1 scores. Especially, the IoU filter provides the most significant improvement.

Table 2. The results of single-camera tracking filtering.

Filters	*IDF*1
baseline	40.99%
speed	44.10%
speed + staytime	51.41%
speed + staytime + IOU	61%

Table 3 presents the effect of the proposed CIR metric. In this experiment, the similarity metric is cosine similarity, and we use the CIR metric for refinement. The top-1 method in the table means the tracklet is directly paired with the highest cosine similarity candidate.

Table 3. Comparison of CIR tracklet association against the top-1 baseline.

Method	*IDF*1	*IDP*	*IDR*
Top-1	42.17%	55.19%	34.13%
CIR	45.80%	64.64%	35.46%

The proposed CIR association scheme can increase the IDF1 score by 3.6% on the validation set. It significantly reduces the number of false positive samples. The number of reduced samples is more than 10,000, which brings significant improvement by our method. It shows that the CIR association scheme can avoid mismatches due to insufficient pairing information. The IDP and IDR of CIR are also improved by the top-1 method.

Table 4 presents the ablation study brought by each TSN module. The first two rows are the results of the previous experiments. Under the same similarity metric *Cosine Similarity*, the proposed CIR outperforms the pairing method by over 3.6%. The baseline only uses fully connected layers to predict the results. With the proposed CIR, the IDF1 can be improved close to 2%. Next, we compare the results by adding random walk in the reranking phase or training phase. IDF1 scores of 50.08% and 50.83% can be obtained, respectively. Among them, the random walk refined results can be used to calculate the loss to achieve more substantial supervision. Therefore, a better IDF1 score is achieved when adding random walk in the training phase.

Table 4. Performance comparison of TSN modules. DE denotes the disentangle module.

Similarity Metric	Association	*IDF*1	*IDP*	*IDR*
Cosine	Top-1	42.17%	−−	−−
Cosine	CIR	45.80%	−−	−−
Baseline	CIR	47.55%	59.40%	39.64%
Baseline + RW reranking	CIR	50.08%	69.03%	36.66%
Baseline + RW training	CIR	50.83%	65.36%	41.58%
Baseline + RW training + DE	CIR	52.68%	71.49%	41.72%
Baseline + RW training + DE + RLL	CIR	53.15%	73.49%	41.62%

Finally, we use the disentangle module (DE) to separate the camera feature and the ranked list loss in the training stage. With the disentangle module, we obtain the result of 53.15%. This result is 11% higher than the cosine and top-1 methods. In summary, our TSN can separate camera features from tracklet features and uses Ranked List Loss (RLL) to limit negative and positive boundaries. Exploiting random walk into the network allows gallery-to-gallery information to be considered during the training. It makes the similarity of the prediction from the model more accurate. The *IDP* goes up to 73.49% when all components are considered. The *IDR* score is relatively close when RW, DE, and RLL are exploited.

Table 5 shows the results of using different samples to train TSN. *Easy* represents that the dataset contains only easy samples, while *Hard* represents that the dataset contains only hard samples. *Merge* represents the mixture of easy samples and hard samples. The results indicate that sampling with an easy sample achieves a result of 48.43%, while using a hard sample can achieve a result of 49.90%. After mixing the hard and easy samples, the model benefits from the cosine similarity and avoids TSN from being biased toward hard samples. The *IDF*1 can be improved to 53.15%. The *IDP* is decreased a little while *IDR* is improved over 5%.

Figure 7 shows the loss curves of training and validation; following the settings, the training converges at five epochs.

Table 5. The results of the data sampling method.

Dataset	*IDF*1	*IDP*	*IDR*
Easy	48.43%	76.81%	35.36%
Hard	49.90%	78.69%	36.54%
Merge	53.15%	76.89%	41.62%

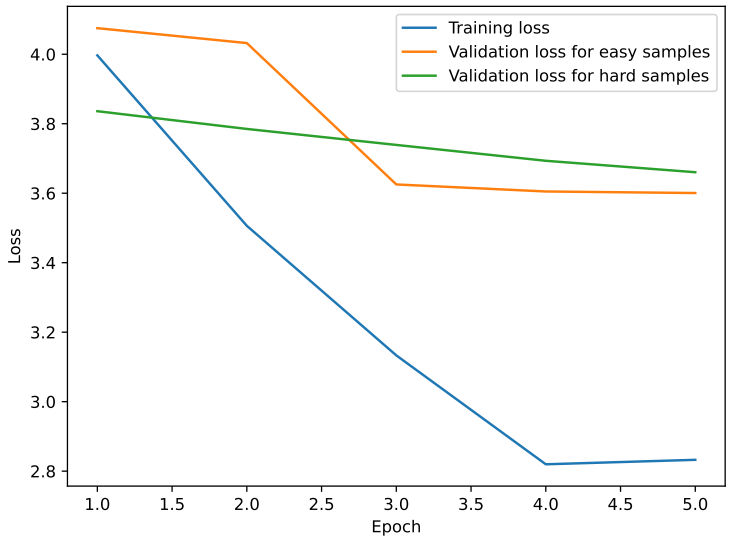

Figure 7. Curves of training loss and validation loss on easy and hard samples.

5. Discussion

The proposed Tracklet Similarity Network (TSN) includes several novel components. Instead of learning image features from a conventional CNN model, Re-Id Block learns the feature representation for each tracklet. Then, Disentangle Block learns the camera-independent feature by exploiting camera loss and ranked list loss functions to achieve feature disentanglement. Lastly, Similarity Measurement Block leverages strong similarities information from the ranking list of tracklets via a random walk scheme rather than using the tracklet with the highest score alone from a query tracklet. The experimental results demonstrate that each component provides incremental improvement for the IDF1 score. The associations of vehicles across cameras from TSN can be further refined by the CIR Tracklet association scheme to achieve multi-camera vehicle tracking. Compared to the recent multi-camera vehicle tracking framework in Table 6, our work is more devoted to developing a multi-camera tracking model and scheme. However, in the front end, the results of single-camera tracking also play an important role in the whole framework, since multi-camera tracking is applied based on the results of single-camera tracking. If the accuracy of vehicle detection and single-camera tracking models can be improved, the overall *IDF*1 can be also significantly increased. In this paper, only baseline methods of vehicle detection and vehicle re-identification are exploited in the multi-target multi-camera tracking framework. For interested readers, please refer to [26] for the state-of-the-art results. In the future, our model can be extended to include the single-camera tracking model as a larger end-to-end tracking system. This can be expected to see higher tracking accuracy.

Table 6. Comparison to the state-of-the-art methods.

Method	*IDF*1
Ren et al. [29]	57.63%
Yang et al. [2]	54.58%
Ours	53.15%

6. Conclusions

The Deep Tracklet Similarity Network is proposed to calculate the similarity between tracklets. Disentangle techniques are applied to reduce the camera information from the tracklet features. In addition, the ranked list loss is adopted and shows significant improvement. The results are refined by the random walk to obtain better tracklet similarities. The Candidates Intersection Ratio scheme is proposed for tracklet association. It leverages the information of the tracklet match list to adjust the tracklet similarity. We integrate tracklets with the same ID into a tree structure in the CIR association scheme. Compared with the existing method, our method significantly improves the IDF1 score by 11% .

Author Contributions: Conceptualization, Y.-L.L., C.-K.C.; methodology, Y.-L.L., H.-T.L., C.-K.C.; software, Y.-L.L., H.-T.L.; validation, Y.-L.L., H.-T.L., C.-K.C.; formal analysis, Y.-L.L., H.-T.L., C.-K.C.; investigation, Y.-L.L., H.-T.L.; resources, Y.-L.L., C.-K.C.; data curation, Y.-L.L.; writing—original draft preparation, Y.-L.L., C.-K.C.; writing—review and editing, H.-T.L., C.-K.C.; visualization, Y.-L.L., H.-T.L.; supervision, C.-K.C.; project administration, C.-K.C.; funding acquisition, C.-K.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Ministry of Science and Technology under the grant 110-2634-F-194-006-.

Acknowledgments: This work was supported by the Advanced Institute of Manufacturing with Hightech Innovations, Center for Innovative Research on Aging Society (CIRAS) and Ministry of Science and Technology under the grant 110-2634-F-194-006-.

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

References

1. Wu, M.; Qian, Y.; Wang, C.; Yang, M. A Multi-Camera Vehicle Tracking System Based on City-Scale Vehicle Re-ID and Spatial-Temporal Information. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Nashville, TN, USA, 19–25 June 2021; pp. 4077–4086.
2. Yang, K.S.; Chen, Y.K.; Chen, T.S.; Liu, C.T.; Chien, S.Y. Tracklet-Refined Multi-Camera Tracking Based on Balanced Cross-Domain Re-Identification for Vehicles. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Nashville, TN, USA, 19–25 June 2021; pp. 3983–3992.
3. Li, P.; Li, G.; Yan, Z.; Li, Y.; Lu, M.; Xu, P.; Gu, Y.; Bai, B.; Zhang, Y. Spatio-temporal Consistency and Hierarchical Matching for Multi-Target Multi-Camera Vehicle Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 16–20 June 2019.
4. Hou, Y.; Zheng, L.; Wang, Z.; Wang, S. Locality Aware Appearance Metric for Multi-Target Multi-Camera Tracking. *arXiv* **2019**, arXiv:abs/1911.12037.
5. Xu, Y.; Liu, X.; Liu, Y.; Zhu, S.C. Multi-view People Tracking via Hierarchical Trajectory Composition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4256–4265. [CrossRef]
6. Bredereck, M.; Jiang, X.; Körner, M.; Denzler, J. Data association for multi-object Tracking-by-Detection in multi-camera networks. In Proceedings of the 2012 Sixth International Conference on Distributed Smart Cameras (ICDSC), Hong Kong, China, 30 October–2 November 2012; pp. 1–6.
7. Hsu, H.M.; Huang, T.W.; Wang, G.; Cai, J.; Lei, Z.; Hwang, J.N. Multi-Camera Tracking of Vehicles based on Deep Features Re-ID and Trajectory-Based Camera Link Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 16–20 June 2019.
8. Qian, Y.; Yu, L.; Liu, W.; Hauptmann, A.G. ELECTRICITY: An Efficient Multi-Camera Vehicle Tracking System for Intelligent City. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 14–19 June 2020.
9. 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 2017, Beijing, China, 17–20 September 2017; pp. 3645–3649. [CrossRef]
10. Basar, T., A New Approach to Linear Filtering and Prediction Problems. In *Control Theory: Twenty-Five Seminal Papers (1932–1981)*; IEEE Press: New York, NY, USA, 2001; pp. 167–179. [CrossRef]
11. Kuhn, H.W. The Hungarian Method for the Assignment Problem. In *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*; Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 29–47. [CrossRef]
12. Wang, Z.; Zheng, L.; Liu, Y.; Li, Y.; Wang, S. Towards Real-Time Multi-Object Tracking. In Proceedings of the Computer Vision—ECCV 2020—16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 107–122. [CrossRef]
13. Ristani, E.; Tomasi, C. Features for multi-target multi-camera tracking and re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6036–6046.
14. Peri, N.; Khorramshahi, P.; Rambhatla, S.S.; Shenoy, V.; Rawat, S.; Chen, J.C.; Chellappa, R. Towards Real-Time Systems for Vehicle Re-Identification, Multi-Camera Tracking, and Anomaly Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 14–19 June 2020.
15. Xu, Y.; Liu, X.; Qin, L.; Zhu, S. Cross-View People Tracking by Scene-Centered Spatio-Temporal Parsing. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 4299–4305.
16. Wen, L.; Lei, Z.; Chang, M.; Qi, H.; Lyu, S. Multi-Camera Multi-Target Tracking with Space-Time-View Hyper-graph. *Int. J. Comput. Vis.* **2017**, *122*, 313–333. [CrossRef]
17. You, S.; Yao, H.; Xu, C. Multi-Target Multi-Camera Tracking With Optical-Based Pose Association. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 3105–3117. [CrossRef]
18. Specker, A.; Stadler, D.; Florin, L.; Beyerer, J. An occlusion-aware multi-target multi-camera tracking system. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 4173–4182.
19. Shen, Y.; Li, H.; Xiao, T.; Yi, S.; Chen, D.; Wang, X. Deep Group-Shuffling Random Walk for Person Re-Identification. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2265–2274. [CrossRef]
20. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015. [CrossRef]
21. Wang, X.; Hua, Y.; Kodirov, E.; Robertson, N.M. Ranked List Loss for Deep Metric Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 5207–5216. [CrossRef]
22. Pan, X.; Luo, P.; Shi, J.; Tang, X. Two at once: Enhancing learning and generalization capacities via ibn-net. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 464–479.
23. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Patt. Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef]

24. Wang, G.; Wang, Y.; Zhang, H.; Gu, R.; Hwang, J. Exploit the Connectivity: Multi-Object Tracking with TrackletNet. In Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, 21–25 October 2019; pp. 482–490. [CrossRef]
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV , Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [CrossRef]
26. Naphade, M.; Wang, S.; Anastasiu, D.C.; Tang, Z.; Chang, M.C.; Yang, X.; Yao, Y.; Zheng, L.; Chakraborty, P.; Lopez, C.E.; et al. The 5th AI City Challenge. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Virtual, 19–25 June 2021.
27. Tang, Z.; Naphade, M.; Liu, M.Y.; Yang, X.; Birchfield, S.; Wang, S.; Kumar, R.; Anastasiu, D.; Hwang, J.N. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 8797–8806.
28. Ristani, E.; Solera, F.; Zou, R.S.; Cucchiara, R.; Tomasi, C. Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. In Proceedings of the Computer Vision—ECCV 2016 Workshops, Amsterdam, The Netherlands, 11–14 October 2016; Part II; Volume 9914, pp. 17–35. [CrossRef]
29. Ren, P.; Lu, K.; Yang, Y.; Yang, Y.; Sun, G.; Wang, W.; Wang, G.; Cao, J.; Zhao, Z.; Liu, W. Multi-Camera Vehicle Tracking System Based on Spatial-Temporal Filtering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Nashville, TN, USA, 19–25 June 2021; pp. 4213–4219.