Real-Time Multi-Vehicle Multi-Camera Tracking With Graph Based Tracklet Features

Transportation Research Record
2023, Vol. XX(X) 1–12

DOI: 10.1177/ToBeAssigned
journals.sagepub.com/home/trr
SAGE

Tuan T. Nguyen[1], Hoang H. Nguyen [2], Mina Sartipi [1] Senior Member IEEE, Marco Fisichella

Abstract

An essential intelligent transportation systems (ITS) application is multi-target multi-camera tracking (MTMCT), where the target's activity is tracked from different cameras. Although the tracking-by-detection scheme is the primary paradigm in MTMCT, the object association information from the video frames is lost. That is mainly because the multi-camera multi-object matching uses the information from the video frames separately. To solve this problem and leverage this association information, we propose an MTMCT framework, where features are built in the form of a graph and a graph similarity algorithm is used to match multi-camera objects. In this paper, we focus on the real-time scenario, where only the past images are used to match an object. Our method achieves an IDF1 score of 0.75 with an FPS of 14

1. Introduction

City-scale intelligent traffic management is getting more effective due to advancements in computer vision research. One important application in intelligent traffic management is vehicle tracking, where spatial, temporal and visual information of the vehicle are integrated to create the vehicle trajectory from different cameras. Multi-Target Multi-Camera Tracking (MTMCT) is an application that extracts the vehicle trajectory on a global scale from cameras located at different locations, as shown in Figure 1. As a result, MTMCT can be used to track vehicles and determine their speed and travel time to optimise traffic flow at the city level. Tracking-by-detection is one of the main paradigms in MTMCT, which is divided into three main components: (1) Object detection, (2) Multi-object tracking (MOT) and (3) Trajectory clustering. As illustrated in Figure 1, objects are detected and labeled with an identifier. MCMCT reidentified "Veh-70" in the following intersection and assigned the same label.

Object detection uses an object detector to extract objects from small images called bounding boxes (bboxes) for each video frame. Then *MOT* tracks the vehicle from the time it enters to the time it leaves the camera view. To match the detected objects within a single camera, MOT uses a tracker such as Deep SORT (*1*). The tracker solves the matching problem by using pairwise object affinities to match detected objects in past frames with detected objects in the current frame. Tracklets containing all bboxes of specific objects in that camera are the result of MOT. Trackers in MOT learn features based on appearance features, such as features from object re-identification (ReID) algorithms.

Figure 1. Multi-target camera tracking (MTMCT) tracks vehicles over cameras.

The objective of Object ReID is to find the same exact vehicle from an extensive gallery set extracted from multiple cameras. Primarily, ReID is based on visual features alone, without using other information such as the license plate number, spatial, and temporal.

Trajectory clustering is the final step of MTMCT, where single-camera tracklets from different cameras are

[1]Center for Urban Informatics and Progress (CUIP) at the University of Tennessee at Chattanooga, 615 McCallie Ave, Chattanooga, TN, 37403, United States
[2] L3S Research Center, Leibniz University of Hannover, Appelstr. 9a, 30167 Hannover, Germany

Corresponding author:
Tuan T. Nguyen, xwz778@mocs.utc.edu

coordinated to track the activity of the object globally. This clustering task is also usually performed using features from an object ReID application. Due to the fact that the traditional ReID algorithm works with separate images of objects, when using tracklets in the trajectory clustering task, the associated information between bboxes may be lost. A given object can be tracked in both offline and online modes. In offline mode, past and future frames are used to track the object, while in online mode, only past frames are used.

In this paper we focus on the online tracking scenario. We propose a framework for the third component of the MTMCT application (trajectory clustering) that achieves good performance with low latency. Our proposed trajectory clustering technique consists of three sub-steps: (a) bbox feature extraction by a Siamese Network, (b) graph-based tracklet features construction, and (c) trajectory matching using a graph similarity algorithm. In other transportation studies, a graph frequently depicts the structure of the road network. However, the graph in this study is formed using Siamase features and Euclidean Distances, and it represents the tracklet feature. Specifically, we create a graph with nodes and edges representing bboxes and the Euclidean distance between their embedding features to solve associated information loss. Then, we use the graph similarity algorithm (SimGNN (*2*)) to compare the graph-based tracklet features and match the vehicle from different cameras.

In summary, our MTMCT framework consists of three primary steps. (1) A vehicle detection algorithm is used to extract the objects as bboxes. (2) The single-camera tracklets are generated. (3) Trajectory clustering that is divided into three sub-steps: (3a) features are extracted for each image using a Siamese Network; (3b) graph-based tracklet features are constructed; and (3c) the graph-based tracklet features and the graph similarity algorithm are used to match objects from different cameras. The following is a summary of our paper's contributions:

- We propose a method for creating graph-based tracklet features that leverage the association information between the bboxes.
- We propose an MTMCT clustering method using graph similarity algorithms.
- Our proposed MTMCT framework achieves an IDF1 score of 0.75 with with an FPS of 14.

The rest of the paper is organised as follows. We first describe the background in Section 2. Background. This is followed by the related work in Section 3. Related Work. Then, our proposed method and architecture are described in Section 4. Methodology. Next, Section 5. Experiments and Results presents our datasets, evaluation metrics, experimental setups and results. Finally, Section 6. Conclusion and Future Work contains our concluding, discussion and future work.

2. Background

Our main contribution is proposing a strategy for creating tracklets in a graph structure using a **Siamase Network** feature and using **graph similarity learning based on Graph Neural Network (GNN)** to group single-camera tracklets into multi-camera tracklets. Therefore, the rest of this section provides an overview of these two topics.

2.1 Siamese Network

It is common for machine learning to be applied to analyze the similarity between two objects. Accordingly, for example, (1) face recognition checks whether an input facial image resembles one stored in the database; (2) question-and-answer websites determine whether a new question is similar to one stored in the database; (3) image search engines display similar images. In this approach, each object is represented as a vector (also known as an embedding). For example: in (3), images are presented as vectors by using the output of a pre-trained convolutional neural network (CNN), then the similarity between the two vectors is calculated to determine the degree to which the two images are similar.

Architecture Two identical subnetworks in the Siamese network, also known as twin networks, are connected at their outputs. Furthermore, the twin networks share not only the same architecture but also the same weights. In parallel, they are responsible for creating vector representations of the inputs. In the case of images, we can use ResNet as a twin network. The Siamese Network can be viewed as a wrapper for twin networks. By measuring similarities between vectors, they assist in the creation of better vector representations.

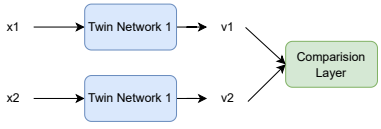


Figure 2. The Siamese Network.

Network Structure x_1 and x_2 in Figure 2 represent the two objects to be compared, and v_1 and v_2 represent their vector representations. In the comparison layer, the architecture is determined by the loss function and the labels associated with the training data. In order to have as much information as possible in the vector representations, the comparison layer is usually designed in a straightforward manner. Below are several common options for setting up the system:

- The cosine similarity between v_1 and v_2 is computed as a real number between -1 and +1. The loss function is the mean square error;
- v_1 and v_2 are concatenated with an absolute difference per element $|v_1 - v_2|$, followed by fully connected

layers and a softmax layer. A cross-entropy loss function is used in this case. Multiclass classification can be accomplished using this option;
- We calculate the Euclidean distance (other distances are also acceptable) between v_1 and v_2, and define the loss function as either a contrastive loss or a triplet loss.

2.2 Graph Similarity Learning Based on Graph Neural Network

A GNN-based similarity learning method performs the similarity learning task in an end-to-end manner while learning graph representations. In order to learn the features of the nodes or graphs in the encoding space of two input graphs, GNN-based graph similarity learning methods first apply multilayer GNNs with weights. Due to the fact that the GNNs of each graph in a pair share weights and/or interact with one another, the learning graphs in a pair may influence each other. A dot product layer or fully connected layer can be used to calculate or predict the similarity values between two graphs based on an output matrix or vector representation provided by the GNN layers. The final step involves combining the similarity scores for all graph pairs and their ground truth labels in a loss function in order to train the model using weighted parameters (*2*–*4*).

A graph similarity learning system should be able to improve accuracy when compared to other state-of-the-art frameworks while maintaining a reasonable computational cost for real-time application. To improve the model's predictability, all properties of the input graphs, including graph topologies and node and edge attributes, are taken into account. The work should adapt Siamese GNNs by incorporating appropriate mechanisms during the learning process with GNNs, and cross-graph interactions should be considered during the learning process for graph representations. Besides, the algorithm may also ignore the input graph's node attributes to reduce computation time.

In summary, the work combines Graph Neural Networks (GNNs) with Convolutional Neural Networks (CNNs) to predict graph similarity. The first step is to train GNNs to represent graphs, and then the learned representations are used to train CNNs to predict similarity scores. As a result, the model becomes a complete end-to-end learning system with additional fully linked layers used to predict similarity values. In this way, the entire process of predicting similarity values can be completed an FPS of 14 while maintaining an acceptable level of accuracy. The online scenario is therefore well suited to this approach.

3. Related Work

Due to the changes in viewpoints, the variations in illumination, and the blind areas among the cameras, tracking multiple targets with multiple cameras is a challenging problem. Several recent studies have investigated the relationship between cameras, including illumination changes, travel times, and entry/exit zones between pairs of cameras.

Since there is considerable variation in illumination between different viewpoints, a brightness transfer function (BTF) is calculated from one camera to another to model these variations. According to Javed et al. (*5*), all BTFs are arranged in a low-dimensional subspace that can be used to compute similarities between appearances. Using the Cumulative Brightness Transfer Function (CBTF), Prosser et al. (*6*) map colours between cameras situated at different locations. Other works (*7, 8*) use the spatial constraints and traffic rules as filters to reduce the searching space in multi-camera trajectory clustering task. Specifically, Liu et al. (*7*) divide the area in the video into crossroad zones, which are used to track the previous and next cameras of a given tracklet. Similarly, Hsu et al. (*8*) propose a traffic-aware module that detects entry- exit zones and is used to match multi-camera tracklets with the camera link model. Besides temporal constraints, some work utilize temporal information in the MTMCT tasks. The model described in (*9*) uses kernel-density estimation to determine the relationships between cameras as multivariate probability densities of spatio-temporal variables and then uses maximum likelihood estimation to incorporate appearance features and these spatial-temporal features

Numerous graph-based models (*10*–*17*) are proposed to deal with MTMCT. A mini-cost flow graph is constructed by Hofmann et al.in (*11*) to complete the data association among cameras in the space of a 3D world. Based on the k-shortest path algorithm, the data association problem in (*13*) is formulated as a constrained flow optimization of a convex problem. In (*15*), a modified multiple hypothesis tracking (MHT) algorithm was developed by Yoon et al. in which branches in track-hypothesis trees represent the trajectory across multiple cameras. Quach et al. (*17*) consider the problem of data association as a link prediction between nodes on a dynamic graph, where the graph vertices are associated with the tracklets, and a self-attention has used the module to embed camera number and temporal information.

Some other works focus on the local features since data associated with tracking systems are usually restricted to a local area. While local area refers to consecutive frames in MOT, it relates to neighboring cameras where the target may be visible in MTMCT. Hou et al. (*18*) propose a locality-aware appearance metric (LAAM), where training data pairs from consecutive frames are sampled in MOT, and training data pairs are selected from consecutive frames of neighboring cameras in MTMCT. Although most MTMCT applications formulate the multi-camera tracklet matching problem as a tracklet-to-tracklet assignment, He et al. (*19*) show that tracklet-to-target assignment (TRACTA) is a better strategy. Specifically, they use the restricted non-negative

matrix factorization (RNMF) algorithm to compute the optimal ID assignment matrix for the tracklet.

In this paper, the best performing MTMCT is used as the baseline described in the section 5. Experiments and Results. In this section, we also present related work on two important components of our framework: MOT and graph similarity.

3.1 Multi-Object Tracking (MOT)

MOT generates single-camera tracklets, the input for multi-camera trajectory clustering tasks in MTMCT. It has been the main trend in the MOT application for many years to use the tracking-by-detection (*20*) paradigm. This paradigm includes three main stages: (1) *image object detection*, (2) *feature extraction*, and (3) *data association*.

Deep learning-based methods have become the main trend in *image object detection* since they can be utilized in small and large networks while preserving speed and accuracy. One of the most common deep learning-based detection algorithms is YOLO (*21*) and its related algorithms such as YOLO-LITE (*22*) for non-GPU computers, Tinier-YOLO (*23*), YOLOv4-tiny (*24*) for real-time applications. Furthermore, Ouyang et al. (*25*) propose a JointDeep method for identifying candidate parts based on the previous detection method, Deformable Part Models (DPM) (*26*). Other works (*27*–*30*) use two-stage R-CNN framework as baseline in object detection architecture. In contrast to detectors that are based on two stages of R-CNN, other works (*31, 32*) use one-stage R-CNN for detecting objects in images.

In the *feature extraction* stage, the detected bboxes are used to extract the features of appearance, motion, and time. For appearance feature, much research has been done to learn image representation, including auto-encoder (*33, 34*), Object re-identification (*35*), feature pyramids (*36*), Transformed-based (*37*) and Siamese Network (*36, 38, 39*). Kalman filters (*1*) and LSTM (*40*) are usually used for features of motion.

Data association group the extracted bboxes by using the extracted features. It usually computes the similarity between extracted features by using various metrics, such as cosine similarity, Euclidean distance, intersection over union, and Siamese Network (*41*). Recent work implements this task using Hungarian algorithms (*1*), dynamic programming (*42*), and reinforcement learning (*43*). Another work proposed by Hsu et al. (*44*) applies post-processing to reduce errors. They use the traffic rule to reduce the ID switch error that splits a single tracklet in a camera into many tracklets.

3.2 Graph Similarity

Based on how to graph similarity or proximity is used in learning, there are two main categories of previous GNN-based work on learning graph similarity, including GNN-CNN mixed models for graph similarity prediction and Siamese GNNs-based graph matching networks.

GSimCNN (*45*) is a model consisting of three stages for pairwise graph similarity prediction. As part of this model, multi-layer graph convolutional networks (GCNs) generate node representations and then compute the inner products of all possible pairs of node embeddings between two graphs derived from different GCN layers. Lastly, multiple independent CNNs and fully linked layers are used to process similarity matrices from different layers to predict the final similarity value.

Ktena et al. (*46*) propose a method for learning graph similarity by employing the siamese graph convolutional neural network. As part of this model, a pair of graphs are considered as inputs, and then a spectral GCN is applied to generate a graph embedding for each graph of input. In the similar work of Ma et al. (*47*), a higher-order Siamese GCN model is proposed to combine the proximity of higher-order nodes with GCNs, and then for the graph similarity learning task, each input graph is applied to higher-order convolutions.

In addition, in recent graph matching networks (*3, 4*) image matching task is implemented as an application in computer vision, where images are transferred to graph topologies. In particular, the graph nodes converted from the input image represent the unary descriptors of the extracted feature points of the image. At the same time, the connections encode the pairwise relationships among these feature points. Furthermore, based on the new graph representation, the image feature matching problem can be reformulated as a graph matching problem.

4. Methodology

The framework **mcmt** (*7*) serves as the foundation for our work. In this paper, steps 1 and 2 in our framework are inspired from **mcmt**, but step 3 is the novelty of our proposed approach. Specifically, our method includes three steps as illustrated in Figure 3. First, in the **Step (1) vehicle detection** step, vehicles are detected as bboxes by Yolo5 from video images. Then, as mentioned in section 3.1 Multi-Object Tracking (MOT), the bboxes are associated to generate corresponding single-camera tracklets in **Step (2) vehicle single camera tracking** model using FairMOT (*48*). We present **Step (3) - trajectory clustering** in three sub-steps. In **Step (3a) feature extraction**, the features of each bbox are learned using the Siamese Network. Following this, a graph-based tracklet feature is constructed based on the graph structure in step **Step (3b) graph-based tracklet feature construction**. Finally, in the **Step (3c) multi-camera tracklets matching** step, a graph similarity algorithm is used to determine the similarity between the tracklets. Using these similarity scores, these tracklets are then matched from different cameras to create tracklets for each vehicle. In the rest of this section, we describe each step in detail.

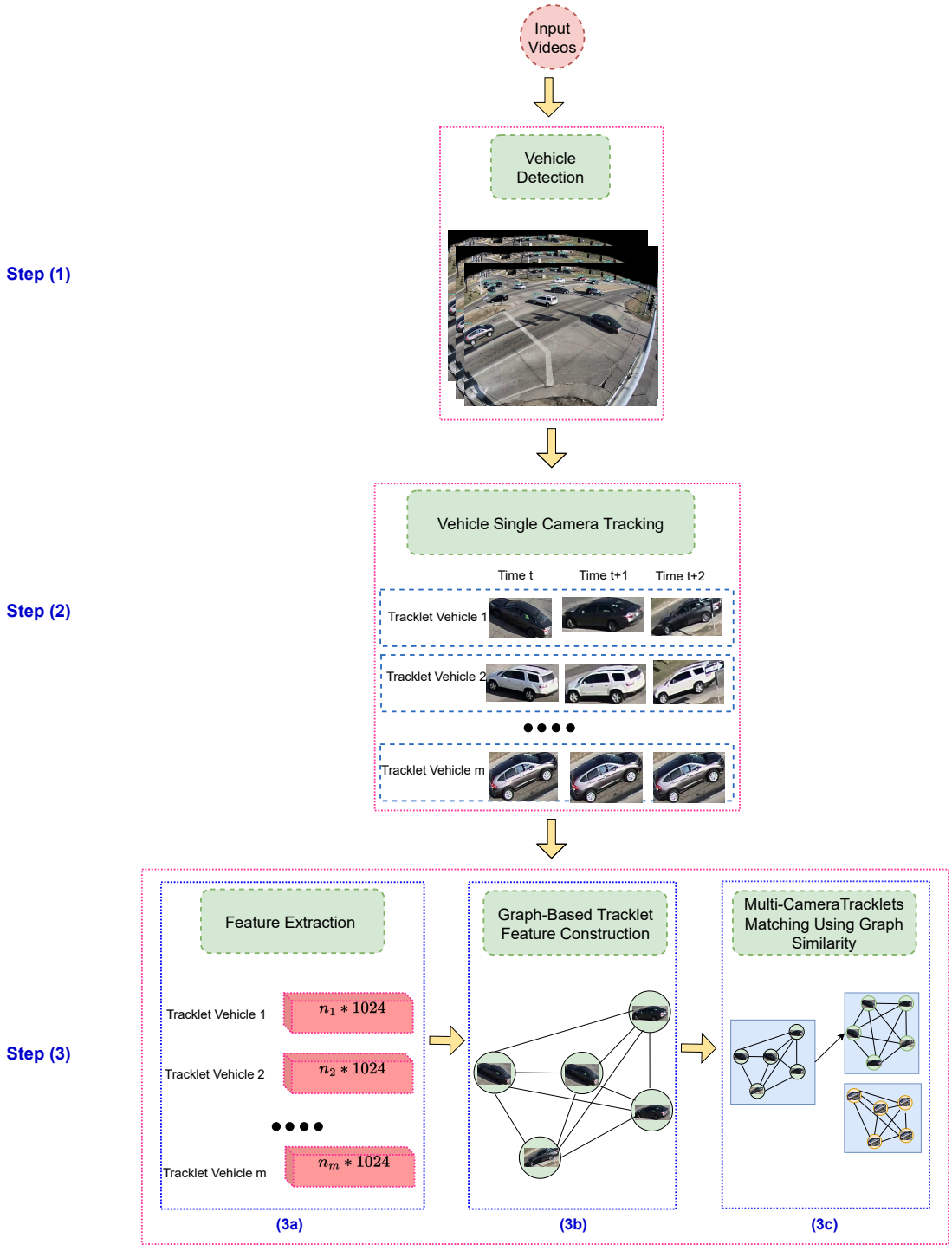


Figure 3. Proposed Architecture. During Step (1) vehicle detection step, the vehicle is first detected. Based on this detection result, single-camera tracklets are generated in Step (2), Vehicle Single Camera Tracking step. In Step (3a), feature extraction, features are learned for each bbox using a Siamese network. Then, tracklet features based on graphs are created in Step (3b) Graph Based Tracklet Feature Construction. In Step (3c), multi-camera tracklets matching, the similarity between tracklets is calculated and used to match single-camera tracklets to generate multi-camera tracklets.

Step 1 - Vehicle Detection

As described in 3.1 Multi-Object Tracking (MOT), the vehicles are detected from the video images as bboxes in this step. Our vehicle detection model is based on YOLOv5 (), which was published in 2020 by Glenn Jocher and

has been investigated in more than 240 research papers. YOLOv5 is developed using the PyTorch framework. A total of 58 open source contributors have contributed to the development of this latest version of the YOLO object recognition model. There are some model configuration files and different versions of this detector, in which we choose the pre-trained model YOLOv5l6 for our implementation. The result is presented as a set of vectors as follows:

$$[t_x, t_y, t_w, t_h, classProb],$$

where t_x, t_y are x and y coordinate of box center in video frame image and t_w, t_h are the weight and height of this box, $classProb$ are the class probability.

Step 2 - Multi-Object Tracking (MOT)

This task is also known as single-camera tracking. To track multiple targets within a single view, we employ the tracking-by-detection paradigm, which combines frame-level detection results into tracklets. In particular, we create tracklets using FairMOT (*48*). In order to estimate the position of objects from noisy measurements, it incorporates a Kalman filtering algorithm as well as a cascade matching algorithm. As a result of the combination of deep visual features and moving information as association criteria, it provides much more information for the association task than simple bounding boxes. As illustrated in Figure 3, the bboxes are associated based on this combination of appearance and moving feature. As a result, the single-camera tracklet containing the triplet vectors is generated for each vehicle as follows:

$$T_{id} = [T_{id,i} : (t_i, b_i, f_i)],$$

where T_{id} is the tracklet with a given id, t_i is the time frame, b_i is a set of vectors of the corresponding bbox as presented in step (1) vehicle detection, and f_i is the appearance feature of the corresponding bbox.

Step 3a - Feature Extraction

The Siamese Network is used in this step to learn the features of each bbox in tracklets. This subsection describes the details of our implemented architecture consisting of two symmetric CNNs with identical structures and parameters, i.e., the Twin Networks from Figure 2. Each CNN is constructed according to the ResNet-50 architecture (*49*) with 50 layers (only the convolutional layers and the fully connected layers are counted). Additionally, to obtain the output with different dimensions, CNN uses a different number of residual units (RUs). Accordingly, CNN contains 3 RUs that output 256 feature maps, 4 RUs that output 512 maps, 6 RUs that output $1,024$ maps, and finally 3 RUs that output $2,048$ maps. ResNet-50 initial parameters are the same as those of the original ResNet-50 trained on $1,000$ classes of the dataset *ImageNet* (*50*). The final structure of each twin Siamese network comprises the ResNet-50 architecture enhanced with two additional layers of d neurons fully connected to the ReLU activation function, which are added at the top.

In the top layer, we obtain a representation of the feature vector v_i for the input image x_i with d dimensions. As a result, different representations (feature vectors) for each input image are generated. Then, the loss used in backpropagation can be calculated via these feature vectors.
Network Training: The weights of all pre-trained layers of the ResNet-50 model are frozen for each Twin Network in order not to affect the weights that the model has already learned. In addition, our own 2 dense layers with d neurons are added to improve the representation. Using Euclidean distance between output image representations, we train the Siamese network by using the standard formulation for *triplet loss* (*51*). This loss function considers triplets (x_a , x_p , x_n) as input. Here x_a is an anchor object, x_p is a positive object (i.e., x_a and x_p belong to the same Vehicle), and x_n is a negative object (i.e., x_a and x_n belong to different Vehicle.Specifically, we randomly select image of vehicle with the same color of x_a). Our goal is to have the vector representation v_a be closer to v_p than it does to v_n. The final formula is:

$$L = max(0,\ m + ||v_a - v_p|| - ||v_a - v_n||).$$

During the training process, $100,000$ triplets were extracted (x_a, x_p, x_n) that served as input to the Siamese Network. The last layer of each CNN outputs vectors (v_a, v_p, v_n) of length d, which serve as representations for the input images. It can be summarized that the Siamese Network is trained simply by entering a triplet of images and backpropagating the losses through each layer.

Step 3b - Graph-Based Tracklet Feature Construction

According to the section 2.2 Graph Similarity Learning Based on Graph Neural Network, we need to construct the tracklet features in a graph structure so that we can utilize the association information in the tracklet and compare the tracklets based on their graph similarity. Specifically, feature vectors (embeddings) from the Siamese Network are represented by nodes, while Euclidean distances between nodes represent edges and a threshold τ determines which edges remain. Then, using a graph similarity algorithm, the similarity between tracklets are computed and used for the next step.

As a result of an empirical tuning proposed in (*52, 53*), it was determined that the distance threshold of τ should be set at 0.5. The human judgment confirmed that 0.5 was the boundary between the same and different images: distances greater than 0.5 were perceived as different pairs, and distances less than 0.5 were perceived as identical pairs.

Crossroad Zone	Camera	Time	Matching
$z_s^i = 1$ or 2	-	$t_e^j < t_s^i$	False
$z_s^i = 3$	$c^j > c^i$	$t_e^j > t_s^i$	False
$z_s^i = 4$	$c^j < c^i$	$t_e^j > t_s^i$	False
$z_s^i = 1$ or 2	-	$t_s^j > t_e^i$	False
$z_s^i = 3$	$c^j > c^i$	$t_s^j < t_e^i$	False
$z_s^i = 4$	$c^j < c^i$	$t_s^j > t_e^i$	False

Table 1. Conflict Table from (7) - In the case of two tracklets that satisfy three conditions of crossroad zone, camera, and time, they do not match.

As a result, a person can recognize that two images are similar but not identical at a distance of 0.5 or more. Additionally, an experiment with different thresholds is conducted in Section 5.4 Experimental Results to confirm our decision. With the aforementioned configuration, tracklet graphs of the CityFlow dataset (54) have an average of 25 nodes and 188 edges. The visualization for a tracklet containing 15 bounding boxes is shown in Figure 4.

Step 3c - Multi-Camera Tracklets Matching Using Graph Similarity

A spatial-temporal filter is applied to increase efficiency before matching. Since all of the cameras are located on the main road as shown in Figure 5, we divide the camera image into four crossroad zones to utilize the spatial information. Figure 6 show an example of crossroad zone in camera 43: 4 colors are used to divide the different zones (white: zone 1; blue: zone 2; green: zone 3; red: zone 4). Specifically, zone 3 and zone 4 are connected to the main road where the cameras are situated. In other words, zone 3 is connected to the next camera (camera 44), whereas zone 4 is connected to the previous camera (camera 42).

As shown in Figure 5, the test scenario includes six cameras with ID numbers ranging from 41 to 46. As a result, we must match tracklets of cameras whose IDs are consecutive. The start/end time $[t_s, t_e]$ and start/end zone $[z_s, z_e]$ of all tracklets T_{id} are extracted before matching. We then filter all tracklet pairs from two cameras with consecutive IDs using the conflict table in table 1. When the three conditions of intersection zone, camera, and time in the conflict table are met, a tracklet pair must not be matched. By using this procedure, the original search space is significantly reduced.

To explain this filtering process with the conflict table, we will use the example of matching tracklets from camera 42 and camera 43. We only need to filter two groups of vehicles from cameras 42 and 43, as shown in Figure 6:

- Group 1: vehicles that move from camera 42 to camera 43.
- Group 2: vehicles that move from camera 43 to camera 42.

For the purpose of filtering these two groups, spatial-temporal filtering is employed. Specifically, as shown in Figure 7, we apply the spatial filtering as follows:

- Group 1: We match the tracklets that end in zone 3 of camera 42 with those that begin in zone 4 of camera 43.
- Group 2: We match the tracklets that end in zone 4 of camera 43 with those that start in zone 3 of camera 42.

Following the spatial filtering, the temporal filter is employed as follows:

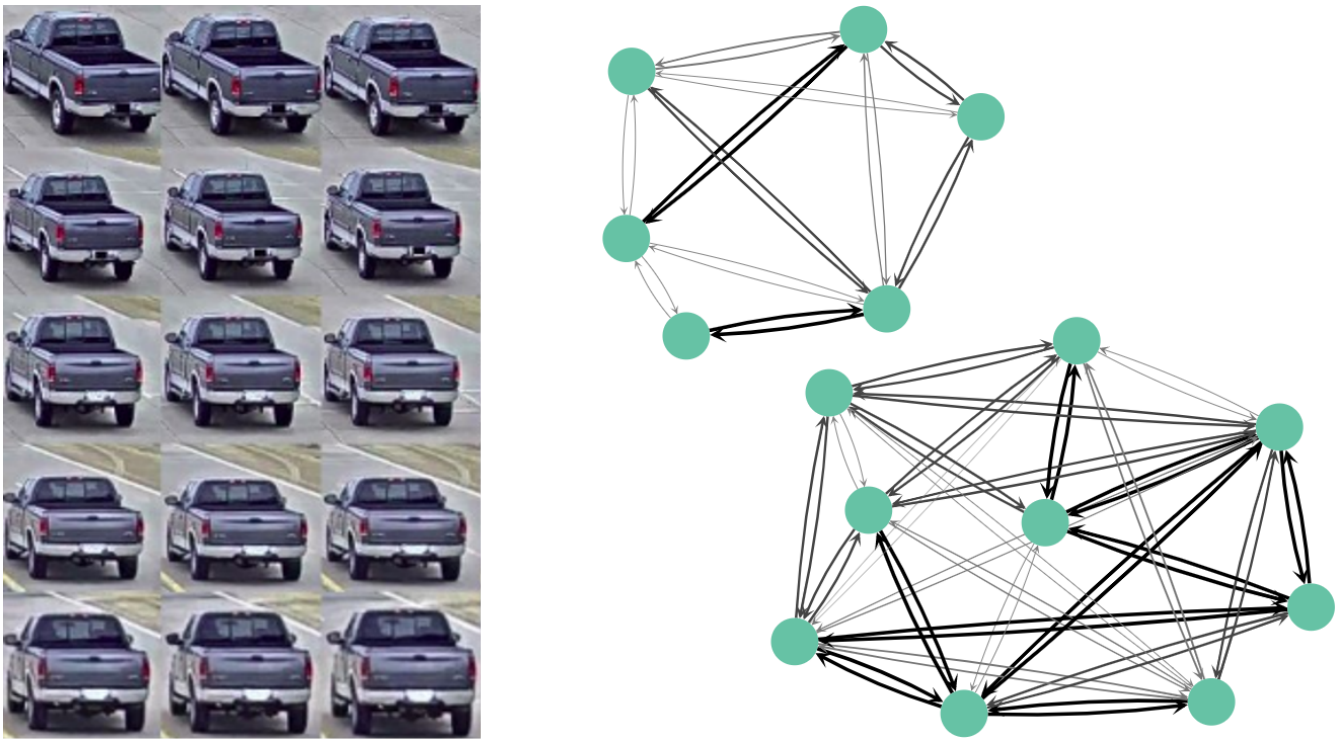
- Group 1: We match the tracklets in camera 42 (ending in zone 3) that have the ending time earlier than the starting time of pairing tracklets in camera 43 (starting in zone 4).
- Group 2: We match the tracklets in camera 43 (ending in zone 4) that have the ending time earlier than the starting time of pairing tracklets in camera 42 (starting in zone 3).

A graph similarity algorithm is applied following the application of the above filters to calculate the similarity scores between graph-based tracklet features. In order to achieve the highest level of performance in matching the input tracklets graphs, the graph similarity algorithm must capture both global (graph-graph) and local (node-node) interactions while also taking into account cross-level interactions between each node of one graph and the other whole graph. Furthermore, since our application runs in real-time, the computational cost must be acceptable. In particular, SimGNN (2), a simplified GNN model that ignores node features in favor of graph topologies, is used to learn graph similarity. Specifically, the model have an FPS of 14 using SimGNN. Additionally, most current state-of-the-art approaches to the MTMCT problem (Vehicle tracking) work offline, not online. Therefore, to the best of our knowledge, we are the first to create an approach for handling Vehicle MTMCT problems with acceptable FPS (14) and high accuracy (achieving an IDF1 score of 0.75).

5. Experiments and Results

5.1 Dataset and Operation System

To conduct the experiment, we used the CityFlow dataset (54), which contains multiple street camera images of the actual scenes in the city. A notable advantage of CityFlow is that it covers different types of streets, including intersections, highways, and road extensions. There are 3.25 hours of traffic videos captured from 40 cameras at ten intersections in both the training and validation sets. The test set consists of 20 minutes of street video from six cameras



(a) Tracklet (b) Graph-Based Tracklet Feature

Figure 4. Graph-Based Tracklet Feature Visualization: (a) A tracklet consisting of 15 bounding boxes collected from various video frames. (b) a graph-based tracklet feature in which nodes are bounding boxes and edges represent the Euclidean distance between nodes; the darker the edges, the greater the distance.

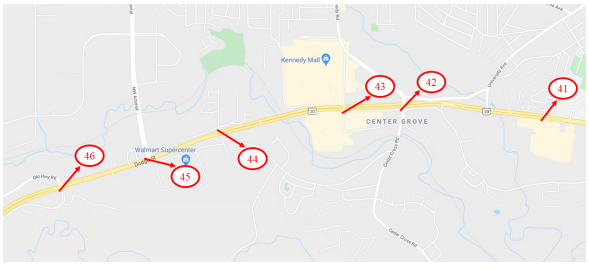


Figure 5. Camera Locations - Testing scenario includes 6 camera from ID 41 to 46.

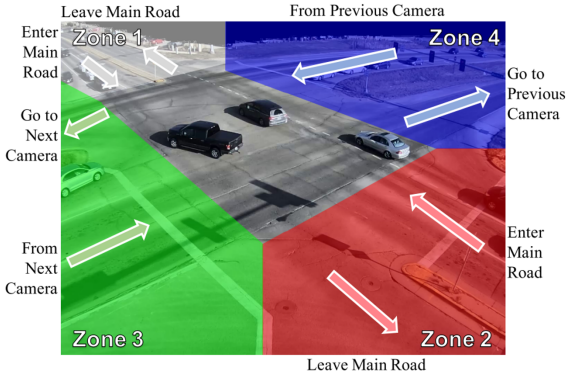


Figure 6. Crossroad Zone Visualization of camera 43 from (*7*). To utilize the spatial information, the video image is divided into four zones. Zone 4 and zone 3 connect to the previous and next camera. Zones 1 and 2 are the exits from the main road.

situated at six intersections. The training experiments are run on 4 Nvidia Tesla 32GB V100s and the inference experiments are run on 1 Nvidia Tesla 32GB V100s.

5.2 Evaluation Metrics

Our method's performance is evaluated using precision, recall, and IDF1 (*55*). IDF1 is defined as the ratio of the number of correctly identified objects to the number of ground truth and average objects. The IDF1 formula is presented below:

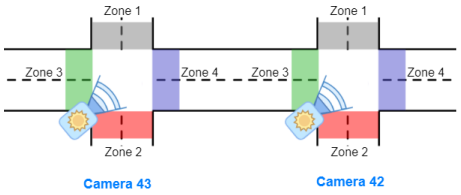


Figure 7. The position of camera 43, camera 42 and their zones. Between cameras 43 and 42, there are two groups of vehicles moving. The vehicles of group 1 move from zone 4 of camera 43 to zone 3 of camera 42, while those in Group 2 moves from zone 3 of camera 42 to zone 4 of camera 43.

$$IDF1 = \frac{2 * TP}{2 * TP + FP + FN}$$

where TP is true positive, FP is false positive, and FN is false negative matching.

5.3 Baselines

As mentioned in section 4. Methodology, **mcmt** is the foundation for our work. However, we do not compare this framework in our experiments since **mcmt** architecture is designed for the offline scenario requiring up to hours for the model training and inference and can not respond in real-time. The results of our study are compared with those of other online tracking algorithms, such as **TADAM** (*56*) and **BLSTM-MTP** (*57*). The **TADAM** model provides a synergistic combination of position prediction and embedding association. Specifically, the prediction aims to focus attention on targets and less on distractions by using attentional modules. As a result of such reliable embeddings, identity awareness perception can be enhanced through memory aggregation. By using a novel multi-track pooling module, the authors of **BLSTM-MTP** aim to solve the issue of simultaneously updating all tracks in memory with a spatial overhead.

5.4 Experimental Results

The performance of our methods has been demonstrated in a number of ablation studies.

To begin with, we analyze the results using two different Siamese Network architectures: ResNet and Efficient Net. According to Table 2, for features of dimension 1024, ResNet achieves a better result, whereas Efficient Net has a faster fps. Our next step is to keep the Siamese Network Backbone as ResNet and increase the dimension of the feature to 2048 and 4096 to determine which is most appropriate for the feature dimension. Based on the results shown in Table 2, it is surprising that the feature with dimension 2048 obtained the best IDF1 value of 0.7521, while the feature with dimension 4096 obtained an IDF1 value of 0.7357. The average FPS of our system is 14, which is suitable for a real-time scenario.

	Method	FPS	IDF1
Baselines	TADAM (*56*)	13.30	0.5347
	BLSTM-MTP (*57*)	8.55	0.6167
Ours	SimGNN - Efficient Net (1024-dimension Feature)	**16.70**	0.6530
	SimGNN - ResNet (1024-dimension Feature)	15.63	0.7316
	SimGNN - ResNet (2048-dimension Feature)	14.03	**0.7521**
	SimGNN - ResNet (4096-dimension Feature)	12.16	0.7357

Table 2. Result on different Siamese Network backbones and feature's dimension based-on SimGNN.

Threshold τ	IDF1
0.4	0.7325
0.5	**0.7521**
0.6	0.7439

Table 3. Result on different Graph Construction thresholds

Similarity Metric	IDF1
Cosine Similarity	0.7443
Manhattan Distance	0.7162
Euclidean distances	**0.7521**

Table 4. Result on different Similarity Metric

Based on an empirical tuning proposed in (*52*, *53*), the distance threshold τ is set to 0.5 in the construction of the graph feature. It is necessary, however, to conduct an ablation study with different thresholds in order to confirm this decision. In particular, we examine the final results using thresholds of $0.4, 0.5$, and 0.6. In these experiments, the backbone is a Siamese Network with a 2048-dimension. As can be seen from Table 3, a threshold of 0.5 yields the best IDF1 of 0.7521, which is in agreement with previous work (*52*, *53*).

All above experiments using Euclidean distances to compare the similarity of graph feature. In addition, we evaluate the the performance of algorithm of with Manhattan Distance and Cosine Similarity. In these experiments, the backbone is a Siamese Network with a 2048-dimension. The result from table 4 show that algorithm with Euclidean distances achieve the best IDF1 of 0.7521. With optimal settings (SimGNN + ResNet + 2048-dimension Feature), the overall training time for the 3.25-hour train video is 187 hours and the total inference time for the 20-minute test video is 14.25 minutes.

Baseline Comparison: Some other online tracking algorithms have also been evaluated, we train TADAM (*56*) and BLSTM-MTP (*57*) model on CityFlow dataset for the comparison. The results of these two algorithms are not comparable since they were developed for pedestrian tracking.

The IDF1 for TADAM is 0.53, while the IDF1 for BLSTM-MTP is 0.61. Perhaps more importantly, the fps for TADAM and BLSTM-MTP are respectively 13.33 and 8.55 which are slower than our algorithm. According to Table 2, our fps is competitive, while the IDF1 is superior to existing methods.

6. Conclusion and Future Work

As part of an intelligent traffic management system, MTMCT is a critical task. Using this technology, one can track the trajectory of vehicles in the city and determine their speed and travel time to optimize traffic flow. A graph-based tracklet feature is created based on the association information between the bboxes of a tracklet. A graph similarity learning algorithm is then used to determine the similarity value, which is used to match the single-camera tracklets from different cameras to form a multi-camera tracklet for each vehicle. This strategy is used for the online scenario in which objects are only associated with previous frames. Since a real-time application requires a fast inference running time, we customize SimGNN to serve as a learning graph similarity algorithm. Thus, our real-time approach achieves a competitive IDF1 score of 0.7521 with an FPS of 14.

In spite of the fact that we are currently focusing on online vehicle tracking, the proposed method can also be applied to pedestrian tracking due to the fact that it takes advantage of body-moment matching information. Specifically, a person's body parts can be used as nodes to create a graph-based structure for that individual. We can then apply a similar technique to match the person from the previous frame with the person in the current frame. As a result, one possible direction for future research is to expand our framework to include pedestrian tracking.

Although all test set cameras in this study are installed along the same road, this does not limit the applicability of this baseline to other, more complex road networks, such as those with intersections, overpasses, etc. In these instances, the application of this study can be simply utilized with a more complex conflict table.

Acknowledgement

This material is based upon work partially supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Award Number DE-EE0009208. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This work was also supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 833635 (project ROXANNE: Real-time network, text, and speaker analytics for combating organized crime, 2019-2022).

Author Contribution Statement

All authors contributed equally in providing critical feedback and helping shape the research, analysis, planning, and development of the evaluation and manuscript. M.F. conceived the original idea and experimental settings and directed the project. T.T.N. and H.H.N. developed the framework and performed the experiments for the evaluation. M.F. and M.S. verified the analytical methods. All authors reviewed the results and approved the final version of the manuscript.

References

1. Wojke, N., A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.
2. Bai, Y., H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 384–392.
3. Jiang, B., P. Sun, J. Tang, and B. Luo. Glmnet: Graph learning-matching networks for feature matching. *arXiv preprint arXiv:1911.07681*.
4. Wang, R., J. Yan, and X. Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3056–3065.
5. Javed, O., K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 2. IEEE, 2005, pp. 26–33.
6. Prosser, B., S. Gong, and T. Xiang. Multi-camera Matching using Bi-Directional Cumulative Brightness Transfer Functions. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2008, pp. 64.1–64.10. Doi:10.5244/C.22.64.
7. Liu, C., Y. Zhang, H. Luo, J. Tang, W. Chen, X. Xu, F. Wang, H. Li, and Y.-D. Shen. City-scale multi-camera vehicle tracking guided by crossroad zones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4129–4137.

8. Hsu, H.-M., Y. Wang, and J.-N. Hwang. Traffic-aware multi-camera tracking of vehicles based on reid and camera link model. In *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 964–972.
9. Javed, O., K. Shafique, Z. Rasheed, and M. Shah. Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding*, Vol. 109, No. 2, 2008, pp. 146–162.
10. Papakis, I., A. Sarkar, and A. Karpatne. Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization. *arXiv preprint arXiv:2010.00067*.
11. Hofmann, M., D. Wolf, and G. Rigoll. Hypergraphs for joint multi-view reconstruction and multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3650–3657.
12. Wen, L., Z. Lei, M.-C. Chang, H. Qi, and S. Lyu. Multi-camera multi-target tracking with space-time-view hyper-graph. *International Journal of Computer Vision*, Vol. 122, No. 2, 2017, pp. 313–333.
13. Berclaz, J., F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 33, No. 9, 2011, pp. 1806–1819.
14. Liu, W., O. Camps, and M. Sznaier. Multi-camera multi-object tracking. *arXiv preprint arXiv:1709.07065*.
15. Yoon, K., Y.-m. Song, and M. Jeon. Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views. *IET Image Processing*, Vol. 12, No. 7, 2018, pp. 1175–1184.
16. Tesfaye, Y. T., E. Zemene, A. Prati, M. Pelillo, and M. Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *arXiv preprint arXiv:1706.06196*.
17. Quach, K. G., P. Nguyen, H. Le, T.-D. Truong, C. N. Duong, M.-T. Tran, and K. Luu. Dyglip: A dynamic graph model with link prediction for accurate multi-camera multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13784–13793.
18. Hou, Y., L. Zheng, Z. Wang, and S. Wang. Locality aware appearance metric for multi-target multi-camera tracking. *arXiv preprint arXiv:1911.12037*.
19. He, Y., X. Wei, X. Hong, W. Shi, and Y. Gong. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing*, Vol. 29, 2020, pp. 5191–5205.
20. Ciaparrone, G., F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, Vol. 381, 2020, pp. 61–88.
21. Redmon, J., S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
22. Huang, R., J. Pedoeem, and C. Chen. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.
23. Fang, W., L. Wang, and P. Ren. Tinier-YOLO: A real-time object detection method for constrained environments. *IEEE Access*, Vol. 8, 2019, pp. 1935–1944.
24. Amrouche, A., Y. Bentrcia, A. Abed, and N. Hezil. Vehicle Detection and Tracking in Real-time using YOLOv4-tiny. In *2022 7th International Conference on Image and Signal Processing and their Applications (ISPA)*. IEEE, 2022, pp. 1–5.
25. Ouyang, W. and X. Wang. Joint deep learning for pedestrian detection. In *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2056–2063.
26. Felzenszwalb, P. F., R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 32, No. 9, 2010, pp. 1627–1645.
27. Ren, S., K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, Vol. 28.
28. Zhang, S., L. Wen, X. Bian, Z. Lei, and S. Z. Li. Occlusion-aware R-CNN: detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 637–653.
29. Noh, J., S. Lee, B. Kim, and G. Kim. Improving occlusion and hard negative handling for single-stage pedestrian detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 966–974.
30. Song, T., L. Sun, D. Xie, H. Sun, and S. Pu. Small-scale pedestrian detection based on topological line localization and temporal feature aggregation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 536–551.
31. Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 2016, pp. 21–37.
32. Duan, K., S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6569–6578.
33. Feng, H., X. Li, P. Liu, and N. Zhou. Using stacked auto-encoder to get feature with continuity and distinguishability in multi-object tracking. In *International Conference on Image and Graphics*. Springer, 2017, pp. 351–361.
34. Ho, K., J. Keuper, and M. Keuper. Unsupervised multiple person tracking using autoencoder-based lifted multicuts. *arXiv preprint arXiv:2002.01192*.
35. Kuma, R., E. Weill, F. Aghdasi, and P. Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–9.
36. Lee, S. and E. Kim. Multiple object tracking via feature pyramid siamese networks. *IEEE access*, Vol. 7, 2018, pp. 8181–8194.

37. He, S., H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang. Transreid: Transformer-based object re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15013–15022.
38. Kim, M., S. Alletto, and L. Rigazio. Similarity mapping with enhanced siamese network for multi-object tracking. *arXiv preprint arXiv:1609.09156*.
39. Lavi, B., M. F. Serj, and I. Ullah. Survey on deep learning techniques for person re-identification task. *arXiv preprint arXiv:1807.05284*.
40. Milan, A., S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI conference on artificial intelligence*. 2017.
41. Liang, Y. and Y. Zhou. LSTM multiple object tracker combining multiple cues. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2351–2355.
42. Ullah, M. and F. Alaya Cheikh. A directed sparse graphical model for multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1816–1823.
43. Ren, L., J. Lu, Z. Wang, Q. Tian, and J. Zhou. Collaborative deep reinforcement learning for multi-object tracking. In *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 586–602.
44. Hsu, H.-M., T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang. Multi-Camera Tracking of Vehicles based on Deep Features Re-ID and Trajectory-Based Camera Link Models. In *CVPR Workshops*. 2019, pp. 416–424.
45. Bai, Y., H. Ding, Y. Sun, and W. Wang. Convolutional set matching for graph similarity. *arXiv preprint arXiv:1810.10866*.
46. Ktena, S. I., S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, Vol. 169, 2018, pp. 431–442.
47. Ma, G., N. K. Ahmed, T. L. Willke, D. Sengupta, M. W. Cole, N. B. Turk-Browne, and P. S. Yu. Deep graph similarity learning for brain data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 2743–2751.
48. Zhang, Y., C. Wang, X. Wang, W. Zeng, and W. Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, Vol. 129, No. 11, 2021, pp. 3069–3087.
49. He, K., X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.
50. Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.
51. Schroff, F., D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 815–823. doi:10.1109/CVPR.2015.7298682. URL https://doi.org/10.1109/CVPR.2015.7298682.
52. Fisichella, M., A. Ceroni, F. Deng, and W. Nejdl. Predicting Pair Similarities for Near-Duplicate Detection in High Dimensional Spaces. In *Database and Expert Systems Applications - 25th International Conference, DEXA 2014, Munich, Germany, September 1-4, 2014. Proceedings, Part II*, *Lecture Notes in Computer Science*, Vol. 8645 (H. Decker, L. Lhotská, S. Link, M. Spies, and R. R. Wagner, eds.). Springer, 2014, pp. 59–73. doi:10.1007/978-3-319-10085-2_5. URL https://doi.org/10.1007/978-3-319-10085-2_5.
53. Fisichella, M. Siamese coding network and pair similarity prediction for near-duplicate image detection. *International Journal of Multimedia Information Retrieval*. doi:10.1007/s13735-022-00233-w. URL https://doi.org/10.1007/s13735-022-00233-w.
54. Tang, Z., M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang. 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*. 2019, pp. 8797–8806.
55. Ristani, E., F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*. Springer, 2016, pp. 17–35.
56. Guo, S., J. Wang, X. Wang, and D. Tao. Online multiple object tracking with cross-task synergy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8136–8145.
57. Kim, C., L. Fuxin, M. Alotaibi, and J. M. Rehg. Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9553–9562.